TECHNISCHE
UNIVERSITÄT
DARMSTADT

Context-Aware Decision Making in Wireless Networks: Optimization and Machine Learning Approaches

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades einer
Doktorin der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation

von

M.Sc. Sabrina Klos (geb. Müller)

geboren am

16.09.1988 in Groß-Gerau

Referentin:	Prof. Dr.-Ing. Anja Klein
Korreferent:	Prof. Dr.-Ing. Bernd Freisleben
Tag der Einreichung:	05. Juli 2019
Tag der mündlichen Prüfung:	29. August 2019

D 17
Darmstadt 2019

The work of Sabrina Klos has been funded by the *Deutsche Forschungsgemeinschaft (DFG)* within the Collaborative Research Center (CRC) 1053 *Multi-Mechanismen-Adaption für das künftige Internet (MAKI)* as part of project B3. (`https://www.maki.tu-darmstadt.de`)

Klos, Sabrina:
Context-Aware Decision Making in Wireless Networks: Optimization and Machine Learning Approaches
Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2019
URN: urn:nbn:de:tuda-tuprints-91761
Tag der mündlichen Prüfung: 29.08.2019

Acknowledgements

First, I would like to sincerely thank my advisor Prof. Anja Klein who gave me the great opportunity to pursue my doctoral studies in her group. She has provided excellent guidance through all the stages of my doctoral studies, always supported me, and encouraged me to conduct research in this very interesting area between engineering and mathematics. I am also very grateful for the opportunities I have been given to participate in conferences and research visits abroad. Also, I would like to thank Prof. Bernd Freisleben for agreeing to be the second referee of this thesis.

I gratefully acknowledge all my collaborators. A special mention goes to Prof. Mihaela van der Schaar. I want to sincerely thank her for hosting me during research visits in her laboratories at the University of California, Los Angeles and at the University of Oxford. Her enthusiasm and her wealth of ideas are very inspiring. I would also like to express my sincere gratitude to Prof. Cem Tekin and Dr. Onur Atan for their help in the area of multi-armed bandits.

All the past and present members of KT have made these last couple of years a great time. Special thanks to Lioba for always helping with administrative matters and caring for "her" PhD students. Many thanks to the old generation of colleagues Daniel, Andrea, Mousie, Fabian, Alex, Mahdi, Alexey, and Hussein, for discussions, mutual support, and simply for all the fun we had. Also, many thanks to the "newies" Tobias, Kilian, Bernd, Laszlon, and Weskley, for their understanding during the months of writing this thesis and for continuing and establishing KT traditions.

I am grateful for the opportunity to conduct research within the collaborative research center MAKI and for the funding received by the Deutsche Forschungsgemeinschaft (DFG). Being a member of MAKI allowed me to meet many great people and I very much enjoyed the many discussions and conversations, especially with Roland, Arash, Allyson, Amr, and Alex.

Next, I would like to thank my friends Maleen and Katrin for their support (not only in proof-reading this thesis) and all the good times we shared. Thanks also to Patrick and Christian, who have become part of the family, for sticking around for so long now. Special mention goes to the 'coffee crew' that made Fridays much more fun.

Finally, I would like to thank my family for their great support, especially after Frieda's arrival. My greatest thank you goes to Philipp and Frieda. Thank you for your patience, your understanding and your love. Without you, writing this thesis would not have been possible. It belongs to you as much as it belongs to me.

Abstract

In future wireless networks, an enormous number of heterogeneous devices will be connected, leading to a dramatic increase in data traffic. At the same time, future applications will have significantly higher requirements with respect to data rates, reliability, and latency. Conventional approaches, which aim at only improving the communication capabilities of wireless networks, will not be sufficient to satisfy the more demanding requirements arising in future. Hence, a paradigm shift is needed. While conventionally perceived as pure communication networks, wireless networks can provide not only *communication resources*, but also *computation*, *caching*, *data collection*, and even *user resources*. Such resources can be part of the network infrastructure and of the wirelessly connected devices and their users. This radically different view on wireless networks as *networks of distributed connected resources* calls for the development of new techniques that jointly consider and leverage different types of resources in order to improve the system performance.

In this thesis, we show that such new techniques that jointly consider and leverage different types of resources require *context-aware decision making*. This is due to the fact that first, resources need to be shared and secondly, trade-offs between different types of resources exist. Thirdly, the optimal resource allocation may depend not only on network conditions, but also on other node-related, user-related or externally given conditions, the so-called *context*. We provide an overview of context-aware decision making by discussing *context awareness*, *architectures of decision making*, and *designs of decision agents*. Designing a context-aware decision-making framework requires to formulate a context-aware system model. In particular, decision agents responsible for resource allocation need to be identified. These agents may be part of a *centralized*, *decentralized* or *hierarchical* architecture of decision making and a suitable architecture needs to be selected. Finally, designing decision agents requires to model and classify the problem to be solved and to develop an appropriate method according to which decision agents take decisions. We emphasize two designs relevant for context-aware decision making in wireless networks, namely, *optimization-based approaches* and *machine-learning-based approaches*, in the latter case specifically the framework of *multi-armed bandits*.

Moreover, in this thesis, we study three candidate techniques for wireless networks that jointly consider and leverage different types of resources, namely, *computation offloading in multi-hop wireless networks*, *caching at the edge of wireless networks*, and *mobile crowdsourcing*. For each technique, we identify a fundamental problem requiring

context-aware decision making, propose a novel framework for context-aware decision making, and solve the problem using the proposed framework.

Computation offloading allows wirelessly connected devices to offload computation tasks to resource-rich servers. This may reduce the devices' task completion times and their energy consumption. Computation offloading hence trades computation resources off against communication resources. In this thesis, for the first time, we study computation offloading in *multi-hop wireless networks*, where wirelessly connected devices assist each other as relay nodes. We identify the fundamental problem of *context-aware computation offloading for energy minimization in multi-hop wireless networks*. We propose a novel model that takes into account channel conditions, computing capabilities of the devices, task characteristics, and battery constraints at relay nodes since the effect of computation offloading on the devices' energy consumption depends on these context factors. Based on this model, we take an optimization-based approach and formulate the considered problem as a *multi-dimensional knapsack problem*, which takes into account that offloading decisions in multi-hop networks are non-trivially coupled as communication resources of relay nodes need to be shared. Finally, we propose a novel *context-aware greedy heuristic algorithm for computation offloading in multi-hop networks*. Based on its centralized architecture of decision making, this algorithm enables a central entity to take offloading decisions using centrally collected context information. We show that despite its centralized architecture, the algorithm has a small communication overhead. Numerical results demonstrate that the offloading solution found by the proposed algorithm on average reduces the network energy consumption by 13% compared to the case when no computation offloading is used. Moreover, the proposed algorithm yields near-optimal results in the considered offloading scenarios, with a maximal deviation of less than 6% from the global optimum.

Caching at the edge allows popular content to be cached close to mobile users in order to serve user requests locally, thus reducing backhaul and cellular traffic as well as the latency for the user. Hence, caching at the edge exploits caching resources in order to save communication resources. In this thesis, we identify the fundamental problem of *context-aware proactive caching for maximizing the number of cache hits under missing knowledge about content popularity*. We introduce a new model for context-aware proactive caching that takes into account that different users may favor different content and that the users' preferences may depend on their contexts. Using a machine-learning-based approach based on *contextual multi-armed bandits* (contextual MAB), we propose a novel *online learning algorithm for context-aware proactive caching*. Based on its decentralized architecture of decision making, this algorithm enables the controller of a local cache to learn context-specific content popularity, which is typically

not available a priori, online over time. The proposed algorithm takes the cache operator's objective into account by allowing for service differentiation. We analyze the computational complexity as well as the memory and communication requirements of the algorithm, and we show how the algorithm can be extended to practical requirements. Moreover, we derive a *sublinear upper bound on the regret* of the algorithm, which characterizes the learning speed and proves that the algorithm converges to the optimal cache content placement strategy. Simulations based on real data show that, depending on the cache size, the proposed algorithm achieves up to 27% more cache hits than the best algorithm taken from the literature.

Mobile crowdsourcing (MCS) allows task owners to outsource tasks via a *mobile crowdsourcing platform* (MCSP) to a set of workers. Hence, MCS exploits user resources for task solving. In this thesis, we identify the fundamental problem of *context-aware worker selection for maximizing the worker performance in MCS under missing knowledge about expected worker performance*. We present a novel model for context-aware worker selection in MCS that can cope with different task types and that explicitly allows worker performance to be a non-linear function of both task and worker context. Using a machine-learning-based approach based on contextual MABs, we propose a new *context-aware hierarchical online learning algorithm for worker selection in MCS*. Based on the proposed hierarchical architecture of decision making, this algorithm splits information collection and decision making among several entities. *Local controllers* (LCs) in the workers' mobile devices learn the workers' context-specific performances online over time. The MCSP centrally assigns workers to tasks based on a regular information exchange with the LCs. This novel approach solves two critical aspects. First, personal worker context is kept locally in the LCs, which reduces communication overhead and preserves the privacy of the workers, who may not want to share personal context with the MCSP. Secondly, the MCSP is enabled to select the most capable workers for each task based on what the LCs learn about their workers' context-specific performances, which are typically unknown a priori. We analyze the computational complexity and derive upper bounds on the local memory requirements of the algorithm and on the number of times the quality of each worker must be assessed. Moreover, we show that the more access to worker context is granted to the LCs, the lower are the communication requirements of the proposed algorithm compared to an equivalent centralized approach. In addition, we derive a *sublinear upper regret bound*, which characterizes the learning speed and proves that the algorithm converges to the optimal worker selection strategy. Finally, we show in simulations based on synthetic and real data that, depending on the availability of workers, the proposed algorithm achieves an up to 49% higher cumulative worker performance than the best algorithm from the literature.

Kurzfassung

In zukünftigen drahtlosen Netzwerken wird eine extrem hohe Zahl an heterogenen Geräten miteinander kommunizieren, sodass der Datenverkehr enorm ansteigen wird. Zudem werden zukünftige Anwendungen signifikant höhere Anforderungen in Bezug auf Datenraten, Zuverlässigkeit und Latenzzeiten aufweisen. Konventionelle Ansätze, die lediglich darauf abzielen, die Kommunikationsfähigkeiten der drahtlosen Netzwerke zu verbessern, reichen nicht aus, um zukünftigen Anforderungen gerecht zu werden. Daher ist ein Paradigmenwechsel nötig. Konventionell werden drahtlose Netzwerke als reine Kommunikationsnetzwerke verstanden. Zukünftig verfügen sie aber neben *Kommunikationsressourcen* in zunehmendem Maße auch über *Rechen-*, *Speicher-*, *Datenerfassungs-* und sogar *Nutzerressourcen*. Solche Ressourcen sind sowohl Teil der Netzwerkinfrastruktur als auch der drahtlos verbundenen Geräte und ihrer Nutzer. Diese fundamental andere Auffassung von drahtlosen Netzwerken als *Netzwerke verteilter, miteinander verbundener Ressourcen* erfordert die Entwicklung neuer Verfahren, die verschiedene Arten von Ressourcen gemeinsam betrachten und einsetzen, um die Performanz drahtloser Netzwerke zu erhöhen.

In dieser Arbeit zeigen wir, dass Methoden zur *kontextbezogenen Entscheidungsfindung* für neue Verfahren, die verschiedene Arten von Ressourcen in drahtlosen Netzwerken gemeinsam betrachten und einsetzen, benötigt werden. Dies liegt daran, dass erstens Ressourcen geteilt werden müssen und dass zweitens zwischen den verschiedenen Arten von Ressourcen abgewogen werden muss. Drittens kann die optimale Ressourcenallokation nicht nur von Netzwerkbedingungen, sondern auch von weiteren Kontextfaktoren abhängen, die zum Beispiel die Knoten, die Nutzer oder externe Gegebenheiten betreffen. Wir geben einen Überblick über kontextbezogene Entscheidungsfindung, indem wir *Kontextbewusstsein*, *Entscheidungsarchitekturen* und *Agentenentwürfe* diskutieren. Zunächst wird zur Erstellung eines Rahmenwerks für kontextbezogene Entscheidungsfindung ein kontextbewusstes Modell des Systems benötigt. Zudem müssen Entscheidungsträger, sogenannte Agenten, bestimmt werden, die innerhalb des Rahmenwerks für die Ressourcenallokation verantwortlich sind. Die Agenten können Teil einer *zentralisierten*, *dezentralisierten* oder *hierarchischen Entscheidungsarchitektur* sein. Zuletzt muss ein Entwurf der Agenten erstellt werden, indem das betrachtete Entscheidungsproblem modelliert und klassifiziert wird, und eine passende Methode entwickelt wird, anhand derer die Agenten Entscheidungen treffen. Relevante Methoden sind insbesondere *Optimierungsansätze* und *Ansätze des maschinellen Lernens*, im letzteren Fall insbesondere das Rahmenwerk des *mehrarmigen Banditen*.

Darüber hinaus untersuchen wir in dieser Arbeit drei Verfahren, die verschiedene Arten

von Ressourcen in drahtlosen Netzwerken gemeinsam betrachten und einsetzen. Diese sind die Auslagerung von Rechenaufgaben (*Computation Offloading*) in drahtlosen Multi-Hop-Netzwerken, das Speichern von Inhalten am Rand des drahtlosen Netzwerks (*Caching at the Edge*) und das Auslagern von Aufgaben an eine große Anzahl von mobilen Nutzern über das Internet (*Mobile Crowdsourcing*). Für jedes dieser drei Verfahren identifizieren wir ein fundamentales kontextbezogenes Entscheidungsproblem und schlagen ein neuartiges Rahmenwerk für kontextbezogene Entscheidungsfindung vor.

Computation Offloading erlaubt es drahtlos verbundenen Geräten, Rechenaufgaben an ressourcenreiche Server auszulagern, was die Bearbeitungszeit der Rechenaufgaben und den Energieverbrauch der Geräte verringern kann. Somit wird mithilfe von Computation Offloading zwischen Rechenressourcen und Kommunikationsressourcen abgewogen. In dieser Arbeit untersuchen wir zum ersten Mal Computation Offloading in *drahtlosen Multi-Hop-Netzwerken*, in welchen drahtlos verbundene Geräte die Daten anderer Geräte im Sinne einer Relaisstation weiterleiten. Wir identifizieren das fundamentale Problem des kontextbezogenen Computation Offloadings mit dem Ziel der Energieminimierung in drahtlosen Multi-Hop-Netzwerken. Wir schlagen ein neuartiges Modell vor, welches die Kanalbedingungen, die Rechenfähigkeiten der Geräte, die Eigenschaften der Rechenaufgaben und die Batteriebeschränkungen der Relaisstationen berücksichtigt, da der durch Computation Offloading erzielte Nutzen von diesen Kontextfaktoren abhängt. Basierend auf dem vorgeschlagenen Modell wählen wir einen Optimierungsansatz und formulieren das betrachtete Problem als ein *mehrdimensionales Rucksackproblem*, welches die nichttrivialen Kopplungen bei der Auslagerung von Rechenaufgaben einbezieht, die sich daraus ergeben, dass die Kommunikationsressourcen der Relaisstationen geteilt werden müssen. Zuletzt schlagen wir einen neuartigen *kontextbezogenen, heuristischen Greedy-Algorithmus für Computation Offloading in drahtlosen Multi-Hop-Netzwerken* vor. Basierend auf einer zentralisierten Entscheidungsarchitektur ermöglicht dieser Algorithmus einem zentralen Agenten, Entscheidungen über die Auslagerung von Rechenaufgaben unter Zuhilfenahme von zentral gesammelten Kontextinformationen zu treffen. Wir zeigen, dass der Algorithmus trotz seiner zentralisierten Architektur einen geringen Kommunikationsaufwand aufweist. Numerische Ergebnisse legen dar, dass das Auslagern von Rechenaufgaben auf Basis des vorgeschlagenen Algorithmus den Energieverbrauch des Netzwerks im Mittel um 13% senkt, im Vergleich zu dem Fall, dass alle Rechenaufgaben lokal von den Geräten berechnet werden. Zudem erzielt der vorgeschlagene Algorithmus, mit einer maximalen Abweichung von unter 6% vom globalen Optimum, nahezu optimale Lösungen.

Mittels *Caching at the Edge* werden populäre Inhalte nah bei den mobilen Nutzern gespeichert, um deren Anfragen lokal zu bedienen, wodurch die Menge an Mobilfunk-

verkehr und die Latenzzeiten der Nutzer reduziert werden. Somit werden mithilfe von Caching at the Edge Speicherressourcen ausgenutzt, um Kommunikationsressourcen zu sparen. In dieser Arbeit identifizieren wir das grundlegende Problem des kontextbezogenen Cachings at the Edge mit dem Ziel der Maximierung der Anzahl an Nutzeranfragen, die durch die Inhalte im Cache-Speicher abgedeckt werden können (*Cache Hits*), unter fehlender a priori Kenntnis der Popularität von Inhalten. Wir stellen ein neues Modell für kontextbezogenes proaktives Caching at the Edge vor, welches einbezieht, dass verschiedene Nutzer verschiedene Inhalte bevorzugen können und dass die Präferenzen der Nutzer von ihren Kontexten abhängen können. Unter Verwendung eines Ansatzes des maschinellen Lernens, basierend auf dem Rahmenwerk des *kontextabhängigen mehrarmigen Banditen*, schlagen wir einen neuartigen *Online-Lernalgorithmus für kontextbezogenes proaktives Caching at the Edge* vor. Auf Basis einer dezentralisierten Entscheidungsarchitektur ermöglicht dieser Algorithmus dem Controller eines lokalen Cache-Speichers, die kontextspezifische Popularität von Inhalten, die typischerweise a priori nicht bekannt ist, online im Laufe der Zeit zu erlernen. Der vorgeschlagene Algorithmus berücksichtigt die Zielvorgaben des Betreibers eines Cache-Speichers, indem die Differenzierung von Services ermöglicht wird. Wir analysieren die Komplexität, den Speicher- und den Kommunikationsbedarf des Algorithmus und zeigen, wie der Algorithmus an praktische Anforderungen angepasst werden kann. Außerdem leiten wir eine *sublineare obere Schranke für den sogenannten Regret* des Algorithmus her, welche die Lerngeschwindigkeit des Algorithmus charakterisiert und beweist, dass der Algorithmus gegen die optimale Inhaltsplatzierungsstrategie konvergiert. Simulationen auf Basis realer Daten zeigen, dass der vorgeschlagene Algorithmus, in Abhängigkeit der Größe des Cache-Speichers, bis zu 27% mehr Cache Hits erzielt als der beste Algorithmus aus der Literatur.

Mobile Crowdsourcing (MCS) erlaubt es Inhabern von Aufgaben, diese Aufgaben mittels einer *Mobile-Crowdsourcing-Plattform* (MCSP) über das Internet an eine große Anzahl von mobilen Nutzern auszulagern. Somit nutzen MCS-Anwendungen Nutzerressourcen zur Aufgabenlösung aus. In dieser Arbeit identifizieren wir das fundamentale Problem der kontextbezogenen Auswahl von mobilen Nutzern in MCS-Anwendungen mit dem Ziel der Maximierung der Arbeitsleistung unter fehlender a priori Kenntnis der zu erwartenden Arbeitsleistungen individueller Nutzer. Wir stellen ein neuartiges Modell für die kontextbezogene Auswahl von Nutzern zur Aufgabenlösung in MCS-Anwendungen vor, welches verschiedenartige Aufgabentypen zulässt, und welches zudem explizit berücksichtigt, dass die Arbeitsleistung eine nichtlineare Funktion sowohl des Aufgabenkontextes als auch des Nutzerkontextes sein kann. Unter Verwendung eines Ansatzes des maschinellen Lernens, basierend auf dem Rahmenwerk des *kontextabhängigen mehrarmigen Banditen*, schlagen wir einen neuartigen *kontextbezo-*

genen hierarchischen Online-Lernalgorithmus für die Auswahl von Nutzern zur Aufgabenlösung in MCS-Anwendungen vor. Auf Basis einer hierarchischen Entscheidungsarchitektur teilt dieser Algorithmus die Datenerfassung und die Entscheidungsfindung unter mehreren Agenten auf. *Lokale Controller* in den mobilen Endgeräten der Nutzer erlernen die kontextspezifischen Arbeitsleistungen der Nutzer online im Laufe der Zeit. Basierend auf einem regelmäßigen Informationsaustausch mit den lokalen Controllern weist die zentrale MCSP den Nutzern Aufgaben zu. Dieser neuartige Ansatz löst zwei kritische Punkte. Zum einen verbleibt der persönliche Kontext der Nutzer lokal, was den Kommunikationsaufwand reduziert und die Privatsphäre der Nutzer schützt, da letztere ihren persönlichen Kontext möglicherweise nicht mit der MCSP teilen möchten. Zum anderen ermöglicht der Ansatz der MCSP, mithilfe der von den lokalen Controllern erlernten kontextspezifischen Arbeitsleistungen der Nutzer, die typischerweise a priori unbekannt sind, für jede Aufgabe die am besten geeigneten Nutzer auszuwählen. Wir analysieren die Komplexität des Algorithmus und leiten obere Schranken für seinen Speicherbedarf und für die maximal benötigte Anzahl an Qualitätsüberprüfungen eines einzelnen Nutzers her. Außerdem zeigen wir, dass je mehr Nutzerkontext die lokalen Controller zur Verfügung gestellt bekommen, desto kleiner wird der Kommunikationsbedarf des vorgeschlagenen Algorithmus im Vergleich zu einem äquivalenten zentralisierten Ansatz. Zudem leiten wir eine *sublineare obere Schranke für den Regret* des Algorithmus her, welche die Lerngeschwindigkeit des Algorithmus charakterisiert und beweist, dass der Algorithmus gegen die optimale Nutzerauswahlstrategie konvergiert. Zuletzt zeigen wir mittels Simulationen auf Basis synthetischer und realer Daten, dass der vorgeschlagene Algorithmus, in Abhängigkeit der Nutzerverfügbarkeit, eine bis zu 49% höhere kumulative Arbeitsleistung erzielt als der beste Algorithmus aus der Literatur.

Inhaltsverzeichnis

Chapter 1
Introduction

1.1 Distributed Connected Resources in Wireless Networks

Recent years have witnessed a tremendous increase in mobile data traffic [Cis17]. This trend was fueled by an increasing number and extended capabilities of wirelessly connected devices. Personal hand-held mobile devices, such as smartphones, laptops, and tablets, have become more and more popular. Equipped with advanced multimedia and computing capabilities and a plenitude of sensors, today's mobile devices are capable of running resource-hungry mobile applications, such as mobile video, which has become a key generator of mobile data traffic, accounting for half of today's global mobile data traffic [Cis17].

Compared to today's networks, future wireless networks are expected to face even larger demands. On the one hand, traditional mobile device usage will become even more ubiquitous. The global number of mobile users is expected to reach 5.9 billion in 2025 [GSM18]. Aside from mobile video, which is expected to generate an even higher percentage of mobile data traffic in the upcoming years [Cis17], new types of computation-intensive and energy-consuming mobile applications are expected to emerge, such as augmented reality and virtual reality applications. These applications will require high data rates and low latency [Qua18]. On the other hand, applications for the *Internet of things* (IoT) are expected to become much more important for customer as well as industry purposes in future [PDG+16]. Examples of IoT applications envisioned in next generation 5G wireless networks comprise smart cities, smart home, smart factories, smart grids, e-health, and the connected car [GEE+16, PDG+16]. In such IoT applications, heterogeneous devices, such as sensors, actuators, robots, vehicles, smartphones, and other machines and objects embedded with sensors or actuators, are wirelessly connected to the Internet, and are thereby enabled to communicate using *machine to machine* (M2M) communications in order to perform application-specific tasks [AFGM+15]. With the rise of IoT applications, vast numbers of heterogeneous devices will communicate in future wireless networks [Eva11], and, depending on the specific application, high data rates, high reliability, and low latency may be needed.

In view of the increasing amount of data traffic, the increasing number of wirelessly connected devices, and the increasing application requirements, satisfying the demands

in future wireless networks in general and allocating resources in particular will become much more challenging. Conventional approaches typically aim at improving the communication capabilities of the networks, i.e., reducing the delay and increasing data rates and spectral efficiency [LCQ16], by adding more spectrum, more cells, and optimizing the allocation of the available *communication resources* with respect to time, frequency, and space. In particular, over the last years, especially spatial resource usage based on *network densification*, but also advanced multiple access techniques like *orthogonal frequency-division multiple access* (OFDMA), and multi-antenna techniques like *multiple-input and multiple-output* (MIMO), have boosted the capacity and average data rate of cellular networks [LCQ16, AZDG16]. However, these technological advances are not sufficient to satisfy the much more demanding requirements arising in future networks [WZZ+17, WCT+14, LCQ16, AZDG16], such that a *paradigm shift* becomes mandatory.

The new paradigm moves from understanding wireless networks as pure communication networks to understanding them as *networks of distributed connected resources* that provide *communication*, *computation*, *caching* [LCQ16, WZZ+17, HYH+16, WHY+18, CHH+18], *data collection* [HZL16], and even *user resources* [RZZS15]. Figure 1.1 shows an example of a wireless network with heterogeneous wirelessly connected devices and different types of resources distributed over the network.

On the one hand, such resources may be part of the network infrastructure. As proposed in the *mobile edge network* architecture, caching and computational resources may be installed at the edge of the network [HPS+15, WZZ+17, MYZ+17]. For example, *mobile edge computing* (MEC) servers might be attached to *macro base stations* (MBSs) and *cloudlets* [HRR+18], i.e., small scale data centers, may be attached to Wi-Fi access points, in order to provide computing services closer to the devices [WZZ+17]. Moreover, storage space may be added to *small base stations* (SBSs), in order to provide local caching services [WCT+14].

On the other hand, such resources are available in the wirelessly connected devices themselves and their users, such as the communication and computation capabilities, storage space and sensor equipment of the devices and the human intelligence of the mobile users. Exploiting these resources, devices and users in a wireless network can become service providers with respect to communication (e.g., data relaying [MMAS+16, WDM+15]), computation (e.g., task processing [LAK17]), caching (e.g., content storing and sharing [FMASK17]), sensing (e.g., data collection and sharing [HZL16]), and human intelligence (e.g., human-based task completion [RZZS15]). Starting from this radically different view on wireless networks, the question arises

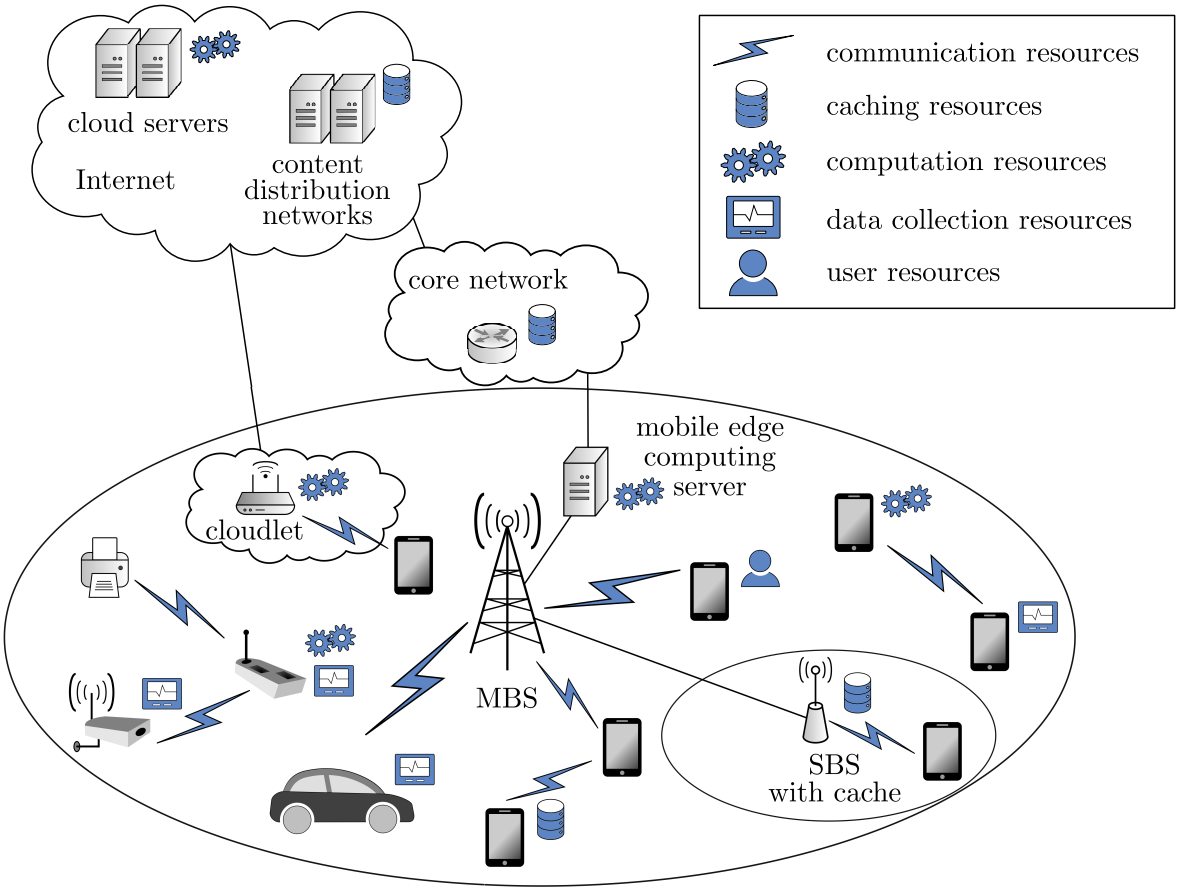


Figure 1.1. Wireless network with distributed, connected resources.

how the different resources available in wireless networks should be exploited and optimally allocated in order to improve the performance of the system with respect to a suitable performance criterion that reflects the requirements of offered services and applications.

1.2 Context-Aware Decision Making in Wireless Networks

The new paradigm calls for the development of new techniques for wireless networks that jointly consider and leverage different types of resources in order to improve the system performance. The goal of introducing such techniques is to allocate the resources available in the wireless network in such a way that the performance of the system is improved with respect to a *performance criterion*, e.g., with respect to the throughput, latency or energy efficiency. The performance criterion may, for example,

be a *global* one, i.e., a network performance criterion, such as the network energy consumption, or it may be a *local* one, i.e., a node performance criterion like individual energy consumption. Resource allocation requires *decision-making during run time*. This is due to the fact that resources are limited and have to be shared among several devices, and moreover, since there may exist trade-offs between different types of resources. More formally, one can think of the different choices (e.g., which resources to use in which way) within a technique that jointly considers and leverages different types of resources, as several available *actions* from which one or several have to be *selected*. Which of the available actions are good choices in turn highly depends on the pre-defined performance criterion. Moreover, whether a selected action is a good choice given a pre-defined performance criterion, may not only depend on the current network conditions, but also on other node-related, user-related or externally given conditions [MSS13, FSK+18]. Conceptually, such conditions may be summarized under the term *context* [DA99, Hen03, MSS13]. Due to the numerous *data collection resources* available in wireless networks, such as the large numbers of sensors in mobile devices, context information is often readily available and may be taken into account for decision making [BWL18]. A detailed introduction to the concept of context will be given in Section 2.2.2.

Clearly, decision making requires one or several entities to act as *decision agents*. In particular, since the available resources and hence the actions within techniques for wireless networks that jointly consider and leverage different types of resources, are distributed over the network, different *architectures of decision making* are possible, namely, *centralized*, *decentralized*, and *hierarchical* architectures [Lun92, KB97]. While in *centralized* architectures, a central entity typically acts as global decision agent based on centrally collected information, in a *decentralized* architecture, several local entities act as local decision agents based on locally collected information. Finally, in *hierarchical* architectures, information collection and decision making may be split up between decision agents at multiple hierarchically organized levels, enabled by an information exchange between them. Each type of architecture has its own advantages and disadvantages, and which type of architecture is suitable highly depends on the specific technique, the specific performance criterion, and the information required for decision making, such as network conditions and other relevant context. The different types of architectures and their characteristics will be discussed in detail in Section 2.2.3.

In any case, decision agents aim at selecting those actions that achieve the defined goal by optimizing the performance criterion. Finding the optimal actions is, however, typically not trivial. How exactly a decision agent selects actions is determined by its *design*. The design of a decision agent essentially depends on how the designer models the problem to be solved by the decision agent. Different approaches differ with respect

to how much of the decision making process is specified a priori by the designer and how the remaining problem is solved by the decision agent [KAC+15]. One possibility, which is often suitable in wireless communications [SNHH15], is to model the problem as an *optimization problem*. Such an optimization problem typically consists of a *utility function*, a set of *constraints*, and a set of *optimization variables* [BV04]. The utility function formalizes the performance criterion of decision making by describing the performance as a function of the selected action. The decision agent may use the utility function to evaluate the performance of selecting an action, while the constraints give restrictions on the actions that may be selected, and the optimization variables describe which actions are selected. Then, the decision agent runs an optimization algorithm in order to search for a (near-) optimal action [BV04].

However, which action gives which performance under a given environment may not be known a priori by the designer, for instance, since underlying parameters may depend on a random process with unknown statistics or may not be easily measurable. In this case, one approach is to use *reinforcement learning* (RL), a type of machine learning [KAC+15]. Here, the designer specifies a set of actions, a set of constraints on these actions, and a performance criterion. Then, the decision agent runs an online learning algorithm that sequentially selects actions and observes their instantaneous performances under different situations in order to learn the performance of the actions under different situations and thereby maximize the cumulative performance over time [SB98, Alp14]. A special case of RL are *multi-armed bandit* (MAB) frameworks [Rob52, ACBF02], which have become a useful tool to tackle problems in wireless communications [MH16, JZR+17]. A detailed overview of the different approaches to design decision agents will be given in Section 2.3.

1.3 Exploiting Distributed Connected Resources

1.3.1 Three Exemplary Techniques

In this thesis, we understand wireless networks as networks of distributed connected resources and we consider how different available resources may be exploited in order to improve the system performance of wireless networks. Specifically, we consider in the sequel how to exploit the following types of resources:

(i) *Computation resources*

(ii) *Caching resources*

(iii) *User resources*

For each type of resource, we take into account its specific interplay with the *communication resources*. In the sequel, for each of the three types of resources (i)-(iii), we present a technique that exploits the particular resource with the help of context-aware decision making. Since context-aware decision making requires access to context information, the three techniques also rely on *data collection resources* that allow to collect the required context information. The following techniques are considered:

(i) *Computation offloading in multi-hop wireless networks*

(ii) *Caching at the edge of wireless networks*

(iii) *Mobile crowdsourcing*

Each of the three techniques has been proposed as a promising candidate to tackle one of the challenges of wireless networks. Below, we will show that each of the three techniques requires context-aware decision making in order to best exploit the resources. In the next sections, we will shortly introduce the three techniques, identify corresponding problems requiring context-aware decision making, and give overviews of the state of the art.

1.3.2 Computation Offloading in Multi-Hop Wireless Networks

Computation offloading allows wirelessly connected devices to offload computation tasks to resource-rich servers for remote computation by transmitting the data required to remotely process the tasks at the servers [KLLB13]. Therefore, computation offloading is a technique that trades *communication resources* off against *computation resources*. The servers enabling computation offloading may be part of different types of infrastructures. For example, *mobile cloud computing* (MCC) allows devices to offload computation tasks to centralized computing platforms in remote servers via the Internet [DLNW13]. In constrast, *mobile edge computing* (MEC) offers computing infrastructure in servers placed at the edge of the wireless networks, e.g., attached

to base stations [HPS+15, WZZ+17, MYZ+17]. Since computing resources are placed much closer to the mobile devices using MEC compared to MCC, employing MEC rather than MCC helps to alleviate the backhaul traffic and to reduce the latency.

Computation offloading may improve the performance of wirelessly connected devices by reducing task completion times, and it also may reduce the devices' energy consumption [KLLB13], thereby facilitating the usage of resource-hungry applications. Indeed, while today's mobile devices are smarter than ever, they still encounter limitations with respect to their battery life and computation capabilities when executing upcoming mobile applications [KLLB13, ASA+14]. This is an issue for mobile users since their most desired feature in mobile devices is a longer battery life, according to several studies [CNN05, You16, You18]. Moreover, many of the machine-type devices communicating in emerging IoT applications dispose of limited processing capabilities [SRI+15, MYZ+17]. Therefore, such devices may profit from computation offloading. Whether computation offloading is beneficial for an individual device in terms of its battery life depends on whether less energy has to be spent for transmitting the task to the server than for local processing. Deciding whether a device should offload its task or not with the goal of *energy minimization* requires not only to trade computation resources off against communication resources by taking into account channel conditions and computing capabilities of a device, but also requires to consider the specific task characteristics [KL10, MN10]. Hence, *context information* is relevant for decision making.

Table 1.1 presents a summary of the most relevant state of the art on decision making for computation offloading. The organization of Table 1.1 and its content will be explained in the sequel. Note that a more detailed review of the state of the art is presented in Section 3.2.

In the past, research has mainly considered computation offloading in *single-hop* networks where devices have a direct connection to a server to utilize its computation resources. One line of literature designs mechanisms to decide whether to offload and which parts of an application to offload, aiming at energy savings or minimization of task completion times from a *single mobile device's point of view* [KL10, MN10, XLL07, WZL12, HWN12, RP03, LMZL16, KLLB13]. Newer works also focus on the *dynamics among several devices* performing computation offloading in single-hop networks, concerning, e.g., traffic induced by computation offloading or competition for shared resources. These works aim at minimizing energy, time, or both energy and time while taking into account the interdependencies between the nodes [GZQL12, MBASK18, NMAS+18, Che15, CLD16].

Table 1.1. Summary of selected related work on decision making for computation offloading.

Network type	**Point of view**	**Optimization criterion**	**Reference**
Single-hop	Device	Min. energy	[KL10] [MN10] [XLL07] [WZL12] [HWN12] [RP03]
		Min. time	[LMZL16]
		Min. energy and time	[KLLB13]
	Network	Min. energy	[GZQL12] [MBASK18]
		Min. time	[NMAS+18]
		Min. energy and time	[Che15] [CLD16]

Since coverage in single-hop networks is limited and transmission may require high power [LH00], it is worthwhile to consider computation offloading in *multi-hop wireless networks*. *Multi-hop communication* allows wirelessly connected devices to communicate directly without the help of a base station, and it allows devices to assist each other as relay nodes. Using multi-hop communication, messages can travel larger distances without needing a base station, which extends the coverage of wireless networks and reduces the required transmission power [LH00, BKK+09]. In the related work, a multi-hop scenario is only considered in a different context of "communication vs. computation," namely in multi-media sensor networks, where data can be compressed at sensor nodes before communicating it to a central entity in a multi-hop fashion [MYM02, TF09].

Compared to single-hop networks, computation offloading in multi-hop networks poses new major challenges. If a device offloads a computation task to the server, other devices may have to serve as relay nodes. However, these relay nodes may have their own computation tasks as well. Moreover, the relay nodes also dispose of limited batteries and hence provide only limited communication resources for task relaying. Finally, several devices may have relay nodes in common, such that they need to share the communication resources provided by the relays nodes. Therefore, offloading decisions are non-trivially coupled in multi-hop networks, which makes it challenging

to decide which devices should offload their tasks.

1.3.3 Caching at the Edge of Wireless Networks

Caching at the edge allows popular content to be cached close to the mobile users in a placement phase in order to locally serve the users' requests for this content in a delivery phase [BBD14b]. Therefore, caching at the edge exploits *caching resources* in order to save *communication resources*. Local caches for caching at the edge could be attached to MBSs and SBSs owned by the *mobile network operator* (MNO). Alternatively, local caches could be part of *wireless infostations* that provide high bandwidth local data communication [GBMY97,IR02,BG14c,BG14a]. Wireless infostations may be installed in public or commercial areas and may use Wi-Fi for local data communication. Owners of wireless infostations may either be content providers aiming at increasing their users' quality of experience or third parties offering caching at infostations as a service to content providers or to the users [BG14a].

Bringing content closer to the mobile users may reduce backhaul and cellular traffic, and it may reduce the latency for the user [WCT+14]. Since a significant amount of mobile traffic is generated by multimedia applications, such as mobile video [Cis17], while at the same time typically only a small number of very popular contents account for the majority of content traffic [BCF+99], caching at the edge may hence enable wireless networks to cope with larger traffic volumes and it may help to satisfy application-specific requirements, e.g., with respect to latency.

Due to the vast amount of content available in multimedia platforms and the limited storage space in local caches at the edge, typically, not all content can be cached locally and a crucial question is hence which content to cache [BG14b]. In order to reduce the load on the macro cellular network as much as possible, the goal is to find a *cache content placement* that *maximizes the number of cache hits*. This requires to use the limited caching resources to cache the most popular content. Which is the most popular content depends on the content popularity distribution [BBD14a].

Table 1.2 presents a summary of the most relevant state of the art on decision making for cache content placement in wireless networks. The organization of Table 1.2 and its content will be explained in the sequel. Note that a more detailed review of the state of the art is presented in Section 4.2.

One line of related work investigates the problem of cache content placement in various caching scenarios in wireless networks under the assumption that the *content popularity*

Table 1.2. Summary of selected related work on decision making for cache content placement in wireless networks.

Content popularity distribution	**Type of learning**	**Diversity in content popularity**	**Reference**
Known	N/A	No	[GMDC13] [SGD+13] [BBD14a]
		Yes	[PT13] [PIST16]
Unknown	N/A	No	[MAN14]
	Offline	No	[BBD14b] [BBZ+15]
	Online	No	[BG14b] [BG14c] [BG14a] [SAT+14]
		Yes	[EBSLa14]

distribution is known a priori [GMDC13, SGD+13, BBD14a, PT13, PIST16]. However, in reality, when caching content at a particular point in time, it may be unclear which content will be requested in future and not even an estimate of the content popularity distribution may be at hand. Therefore, a second line of literature investigates cache content placement *under missing knowledge about content popularity*. If the popularity distribution is *unknown* a priori, the controller of a local cache may either apply a worst-case approach by trying to optimize cache content with respect to worst-case request arrivals [MAN14]. Alternatively, the controller of the local cache needs to *learn* the content popularity distribution [BBD14b, BBZ+15, BG14b, BG14c, BG14a, SAT+14, EBSLa14]. In this case, methods from *machine learning* are leveraged to learn content popularity, using either *offline* or *online* methods. Using offline learning approaches, content popularity is learned during a training phase. Using online learning approaches, content popularity is learned during run time, such that adaptation to varying content popularities is possible.

Related work may further be divided into two groups regarding whether *diversity in content popularity across the user population* is taken into account. Some related works assume that there exists one global popularity distribution and that all user requests follow this distribution. However, it has been shown that the local content popularity at a local cache is not necessarily the same as the global content popularity monitored

by the global multimedia platform [GALM07,ZSGK09,BSW12]. This is due to the fact that there is diversity in content popularity across the user population, i.e., *different* users may favor *different* content. Hence, the controller of a local cache should learn the local content popularity for a *proactive* cache content placement. However, since the set of mobile users connected to a local cache at the edge of the wireless network changes over time, also the local content popularity may vary according to the preferences of the mobile users connecting to the local cache over time. Therefore, for a truly proactive cache content placement, the controller of a local cache needs to take into account the diversity in content popularity across the local user population when learning content popularity. However, only few related works take such diversity in content popularity across the local user population into account for cache content placement.

Among the related works, none takes into account that the users' content preferences may depend on their *contexts*, such as their location [BSW12], personal characteristics (e.g., age [MS10], gender [HL05], personality [RGZ11], mood [Zil88]), or their devices' characteristics [ZGC+14]. However, acknowledging that content popularity depends on the users' contexts, cache content placement needs to be *context-aware*, in order to adapt to the preferences of mobile users with different contexts.

Moreover, none of the literature takes into account that cache content placement should reflect the *cache operator's specific objective*. Since an operator may want to offer *service differentiation* to its customers (e.g., by optimizing cache content according to different prioritization levels [KLAC03, LAS04]), cache content placement should not only allow for cache hit maximization, but also incorporate the operator's perspective by allowing for service differentiation.

1.3.4 Mobile Crowdsourcing

Mobile crowdsourcing (MCS) allows *task owners* to outsource their tasks via an intermediary *mobile crowdsourcing platform* (MCSP) to a set of mobile users, so-called *workers*, who may complete assigned tasks [RZZS15]. Hence, MCS is a technique that exploits *user resources* by leveraging human intelligence for task solving.

While earlier *crowdsourcing* (CS) systems (e.g., Amazon Mechanical Turk[1]) were mainly web-based [DRH11], today, MCS platforms have become increasingly popular, probably due to the large and ever increasing number of mobile devices [Cis17] and the growing intensity of mobile device usage [eMa18]. Tasks in MCS systems may, for

[1] https://www.mturk.com

example, require the mobile users to use their mobile devices in the physical world (e.g., photography tasks), possibly even with respect to certain spatial constraints, the latter being called *spatial* CS [ZH16]. Other tasks are virtual tasks (e.g., image annotation, sentiment analysis), possibly intractable for machine computation, that require human intelligence for their solution. Such virtual tasks are often *non-spatial*, i.e., they do not require the workers to be at a certain location in order to complete the task. While non-spatial tasks could as well be completed by users of static devices as in web-based CS, emerging MCS applications for non-spatial tasks (e.g., MapSwipe[2], the GalaxyZoo app[3], or commercial ones as Spare5[4] or Crowdee[5]) exploit that online mobile users complete such tasks anytime and anywhere on the go. MCS is hence a technique that may enable different stakeholders, e.g., network operators, e-commerce companies, or even the mobile users themselves, to leverage resources of (other) mobile users.

Since different mobile users may have different interests and capabilities, not all mobile users may be equally suitable to complete a given task [GS14]. Hence, MCS requires an appropriate assignment of workers to tasks. In order to achieve the best possible outcome for a task owner, the limited budget of the task owner should be used wisely, by selecting those workers that *maximize the performance* on the given task [TTSRJ14].

Table 1.3 presents a summary of the most relevant state of the art on decision making for CS systems. The organization of Table 1.3 and its content will be explained in the sequel. Note that a more detailed review of the state of the art is presented in Section 5.2.

Literature on CS considers two modes of assigning tasks to workers [KS12]. In the *worker selected tasks* (WST) mode, workers autonomously select tasks from a list. This simple mode, which is often used in practice (e.g., on Amazon Mechanical Turk), has the advantage that workers automatically select tasks they are interested in. However, the WST mode can lead to suboptimal task assignments since it may be difficult for workers to find interesting tasks [CHMA10] and, moreover, unpopular tasks might remain unassigned. Literature on WST mode [GWG+16, AVC11] combines the mode with personalized *task recommendation* (TR) [GS14] to ensure that workers find appropriate tasks.

In the *server assigned tasks* (SAT) mode, the MCSP aims at centrally matching workers and tasks in an optimal way, e.g., to maximize the number of task assignments, taking

[2]https://mapswipe.org/
[3]https://www.galaxyzoo.org/
[4]https://app.spare5.com/fives
[5]https://www.crowdee.de/

Table 1.3. Summary of selected related work on decision making for crowdsourcing systems.

Task assignment mode	**Worker performance**	**Type of learning**	**Context-specific performance**	**Worker context protected**	**Reference**
Worker selected tasks (WST)	Unknown	Offline	Yes	Yes	[GWG+16]
		Online	No	N/A	[AVC11]
Server assigned tasks (SAT)	Known	N/A	No	No	[KS12] [TSK15]
			Yes	Yes	[TGFS17]
	Unknown	Offline	No	N/A	[SC17] [ZC17]
		Online	No	N/A	[HV12] [TTSRJ14] [HZL16]
			Yes	No	[uHC14]

possible task budgets into account. The assignment of workers to tasks is typically based on task and worker information gathered regularly at the MCSP. Related work using the SAT mode often either assumes that workers always accept assigned tasks or that the *workers' performances are known in advance* (e.g., in terms of acceptance rates and quality) [KS12, TSK15, TGFS17]. However, in reality, acceptance rates and quality are typically not known beforehand and therefore have to be learned by the MCSP. A second line of related work therefore considers the matching of workers and tasks in various CS scenarios *under missing knowledge about worker performance* in terms of acceptance rates or quality and proposes *machine-learning-based* approaches, some of them using offline [SC17, ZC17] and others online [HV12, TTSRJ14, HZL16, uHC14] learning.

A worker's performance in terms of acceptance rate and the quality of completed tasks may depend not only on the specific *task*, but also on the *worker's current context*, such as the worker's location or the time of day [GS14]. A worker's context may change quickly, which is especially relevant for MCS applications with non-spatial tasks since workers may complete such tasks anytime and anywhere. Among the discussed related work, only few incorporate such *context-specific worker performance*.

Moreover, in the SAT mode as well as in the WST mode with personalized TR, the

workers are typically required to regularly share their current contexts (e.g., their positions) with the MCSP. This may, on the one hand, require a large communication overhead and, on the other hand, it may be a privacy concern for workers [TGFS17, GWG+16]. *Protecting personal worker context* due to overhead or privacy reasons (i.e., keeping it completely locally, or sharing only generalized context information with the MCSP) has only been taken into account by few previous works.

1.4 Open Issues

In this section, open issues with respect to the three decision-making problems discussed in Sections 1.3.2 – 1.3.4 are summarized.

Computation Offloading Computation offloading has so far not been considered in multi-hop networks, which, compared to single-hop networks, may extend coverage and reduce required transmission power. Since communication resources of relay nodes need to be used and shared for task offloading, offloading decisions are non-trivially coupled in multi-hop networks. In this regard, the following questions arise:

1. How to formulate a general model for context-aware computation offloading in wireless multi-hop networks?
2. How to decide in a wireless multi-hop network which devices should offload their tasks such that the sum energy spent in the overall network for communication and computation is minimized while taking into account the energy constraint in each device? How can information about task context be exploited for decision making?

The underlying optimization problem needs to be analyzed for complexity, and conditions may be derived under which computation offloading in multi-hop networks is beneficial.

3. What is the computational complexity of the optimization problem?
4. Under which conditions is computation offloading beneficial in multi-hop networks?

Finally, the performance of the proposed algorithm should be tested and its computational complexity and overhead need to be studied.

5. How well can the proposed algorithm approximate the optimal solution?
6. What is the complexity and what is the overhead of the proposed algorithm?

Caching at the Edge of Wireless Networks A cache content placement algorithm has not been proposed so far that jointly (i) learns which content to store proactively in a local cache at the edge of the wireless network online under missing a priori knowledge about local content popularity such that the average number of local cache hits is maximized over time, while (ii) allowing for diversity in content popularity across the user population, (iii) taking into account the dependence of the users' preferences on their contexts, and (iv) including the operator's specific objective by supporting service differentiation. Hence, this creates the following questions:

7. How to formulate a model for context-aware proactive caching in a local cache at the edge of the wireless network that (i) takes into account that content popularity may vary across the user population, (ii) considers that the users' preferences depend on their contexts, and (iii) includes the operator's requirements in terms of service differentiation?
8. How to decide online without a priori knowledge about content popularity which content from a large file library to store proactively in a local cache at the edge of the wireless network such that the average number of local cache hits is maximized while taking into account the limited cache size and exploiting local information about user context? How to ensure that cache content placement meets the operator's requirements for service differentiation?

The complexity and overhead of the proposed algorithm need to be investigated and its adaptability to additional practical requirements needs to be proven.

9. What is the complexity and what is the overhead of the proposed algorithm?
10. How well is the proposed algorithm adaptable to practical requirements?

Moreover, the performance of the proposed algorithm should be studied analytically and numerically.

11. How well does the proposed algorithm approximate an orcale-based optimal solution, which would require a priori knowledge about content popularity?

12. How well does the proposed algorithm perform compared to conventional algorithms which either do not learn or which do not exploit context information?

Mobile Crowdsourcing (MCS) A worker selection algorithm for MCS with non-spatial tasks has not been proposed so far that jointly (i) learns online which workers to select for each task under missing a priori knowledge about worker performance in terms of acceptance rate and quality such that the average worker performance is maximized over time, while (ii) allowing for different task types, (iii) taking into account that the worker performance may depend in a possibly non-linear fashion on both task and worker context, and (iv) protecting personal worker context locally in order to keep the communication overhead small and to ensure the workers' privacy. Therefore, the following questions arise:

13. How to formulate a model for context-aware worker selection in an MCS application that allows for different task types and takes into account that worker performance may vary and depend in a possibly non-linear fashion on both task and worker context?

14. How to decide online in an MCS application with non-spatial tasks without a priori knowledge about worker performance which workers from a large set to select such that the average worker performance is maximized over time, while taking into account limited task budgets, task and worker context information and the possibly non-linear relationship between worker performance and context, without giving the central MCS platform access to the worker context?

The computational complexity and the overhead of the proposed algorithm need to be investigated.

15. What is the complexity and what is the overhead of the proposed algorithm?

In order to demonstrate the performance of the proposed algorithm, analytical guarantees and numerical tests are needed.

16. How well can the proposed algorithm approximate an orcale-based optimal solution that would require a priori knowledge about worker performance?

17. How well does the proposed algorithm perform compared to conventional algorithms which either do not learn or which learn in a simpler fashion?

1.5 Contributions and Thesis Overview

This section gives an overview of the thesis and summarizes the main contributions addressing the open issues discussed in Section 1.4. In the following, the contents of each chapter are briefly described, along with the main contributions presented in each of them.

Chapter 2 provides an overview of context-aware decision making in wireless networks. First, an overview of the components of a context-aware system model is given. Then, the context model is discussed by giving a short introduction to context awareness. Moreover, different architectures of decision making and their characteristics are discussed. Finally, different designs of decision agents and corresponding methods for decision making are discussed, with an emphasis on optimization-based approaches and machine-learning-based approaches using *multi-armed bandit* (MAB) models, two specific types of approaches relevant for this thesis.

In Chapter 3, the problem of context-aware computation offloading for energy minimization in multi-hop wireless networks is studied, giving answers to the Questions 1-6 by the following contributions:

1. We propose a general model for context-aware computation offloading in multi-hop wireless networks. The model is applicable to any topology of a multi-hop network in which a resource-rich server may be reached via an *access point* (AP). The model is compatible with different types of infrastructures for computation offloading, such as *mobile cloud computing* (MCC) and *mobile edge computing* (MEC).

2. We use a centralized architecture of decision making and take an optimization-based approach. Specifically, we formulate the network energy minimization problem as an *integer linear programming* (ILP) problem and propose a context-aware greedy heuristic algorithm for computation offloading in multi-hop networks. Using this algorithm, a central entity may take offloading decisions based on centrally collected information about network conditions and task context.

3. We prove the equivalence of the energy minimization problem to a multi-dimensional knapsack problem and thereby derive the complexity of the optimization problem.

4. Based on analytical and numerical evaluation, we derive conditions with respect to the topology, system parameters, and task context under which computation offloading in multi-hop networks is beneficial.

5. We find in numerical simulations that the proposed context-aware greedy heuristic algorithm yields near-optimal results under various network settings and task contexts.

6. We study the computational complexity of the proposed context-aware greedy heuristic algorithm and the overhead of the proposed centralized architecture of decision making with respect to its communication requirements.

Chapter 4 addresses the problem of context-aware caching at the edge for cache hit maximization, and answers Questions 7-12 by the following contributions:

7. We propose a model for context-aware proactive caching in a local cache at the edge of the wireless network. The model explicitly allows different content to be favored by different users and includes that content popularity depends on the user's context.

8. We use a decentralized architecture of decision making and take a machine-learning-based approach. Based on a contextual MAB framework, we present an online learning algorithm for context-aware proactive caching that incorporates diversity in content popularity across the user population, takes into account the dependence of the users' preferences on their contexts, and supports service differentiation. Using this algorithm, the controller of a local cache can learn context-specific content popularity online by regularly observing context information of connected users, updating the cache content, and observing cache hits subsequently.

9. We study the computational complexity of the proposed context-aware proactive caching algorithm and its overhead in terms of memory and communication requirements.

10. We show possible extensions of the proposed context-aware proactive caching algorithm. Specifically, we consider its combination with multicast transmissions, the incorporation of caching decisions based on user ratings, the inclusion of asynchronous user arrivals, and the extension to multiple local caches.

11. We analytically bound the loss of the proposed context-aware proactive caching algorithm compared to an oracle that has a priori knowledge about content popularity. We derive a sublinear upper regret bound, which characterizes the learning speed and proves that the proposed algorithm converges to the optimal cache content placement strategy that maximizes the expected number of cache hits.

12. We numerically evaluate the performance of the proposed context-aware proactive caching algorithm based on a real world data set. A comparison shows that by exploiting context information in order to proactively cache content for currently connected users, the proposed algorithm outperforms reference algorithms.

Chapter 5 investigates the problem of context-aware worker selection for performance maximization in mobile crowdsourcing (MCS) with non-spatial tasks, and answers Questions 13-17 by the following contributions:

13. We propose a model for context-aware worker selection in an MCS application. The model allows different task types by using the concept of task context to describe the features of a task. The model describes worker performance as a possibly non-linear function of the task context and of the worker context.

14. We use a hierarchical architecture of decision making and take a machine-learning-based approach based on a contextual MAB framework. We propose a context-aware hierarchical online learning algorithm for worker selection in MCS applications with non-spatial tasks. The algorithm learns online without requiring a training phase. By adapting and improving the worker selection over time, the algorithm can hence achieve good results already during run time. The proposed algorithm is split into two parts, one part executed by the MCSP, the other part by *local controllers* (LCs) located in each of the workers' mobile devices. An LC learns its worker's performance online over time, by observing the worker's personal contexts and her/his performance. The LC learns from its worker's contexts only locally, and personal context is not shared with the MCSP. Each LC regularly sends performance estimates to the MCSP. Based on these estimates, the MCSP takes care of the worker selection. This hierarchical coordination approach enables the MCSP to select suitable workers for each task based on what the LCs have previously learned.

15. We study the computational complexity of the proposed context-aware hierarchical online learning algorithm and its overhead in terms of local memory and communication requirements. Moreover, we analyze how many times the performance of each worker has to be observed. Keeping this number low is crucial

since observing worker performance requires quality assessments, which may be costly.

16. By establishing an analytical upper regret bound, we provide performance guarantees for the learned worker selection strategy and prove that the proposed context-aware hierarchical online learning algorithm converges to the optimal worker selection strategy.

17. We numerically evaluate the performance of the proposed context-aware hierarchical online learning algorithm based on synthetic as well as real data using different worker performance models. A comparison shows that by exploiting context information for worker selection, the proposed algorithm outperforms reference algorithms.

Finally, the main conclusions of this thesis and a brief outlook on future research directions are presented in Chapter 6.

Chapter 2

Context-Aware Decision Making in Wireless Networks

2.1 Introduction

Along with the new paradigm of understanding wireless networks as networks of distributed connected resources, new techniques are envisioned that jointly consider and leverage different types of resources in order to improve the system performance. In order to optimize resource usage, these techniques require *context-aware decision making* [BWL18, MSS13], as motivated in Section 1.2. A context-aware decision-making framework for such a technique essentially consists of the following two parts:

(i) A *system model* needs to be formulated, consisting of five components, of which an overview will be given in Section 2.2.1. In particular, to allow for context-aware decision making, the designer needs to define a *context model*, specifying which context is needed for decision making and which sources should acquire the context by using their data collection resources [Hen03, PZCG14, MSS13]. Moreover, one or several adequate decision agents need to be identified and an appropriate *architecture of decision making* needs to be designed within which the decision agents are responsible for decision making [Lun92, KB97, FCGS02].

(ii) The *decision agents need to be properly designed*, which requires to model and classify the problem to be solved by the decision agents and to develop an appropriate method according to which the decision agents take decisions [KAC+15, SNHH15, JZR+17, MH16].

Context-aware decision making for new techniques that jointly consider and exploit different resources in wireless networks can be understood as *an interaction between decision agents and the environment* [KAC+15, SB98]. By designing (i) the system model and (ii) the decision agents, the specific properties of this agent-environment interaction are determined. Figure 2.1 shows an illustration of a general agent-environment interaction and connects the components of a context-aware decision making framework with this general agent-environment interaction. More specifically, Figure 2.1 shows a set of decision agents interacting with the environment by taking actions based on

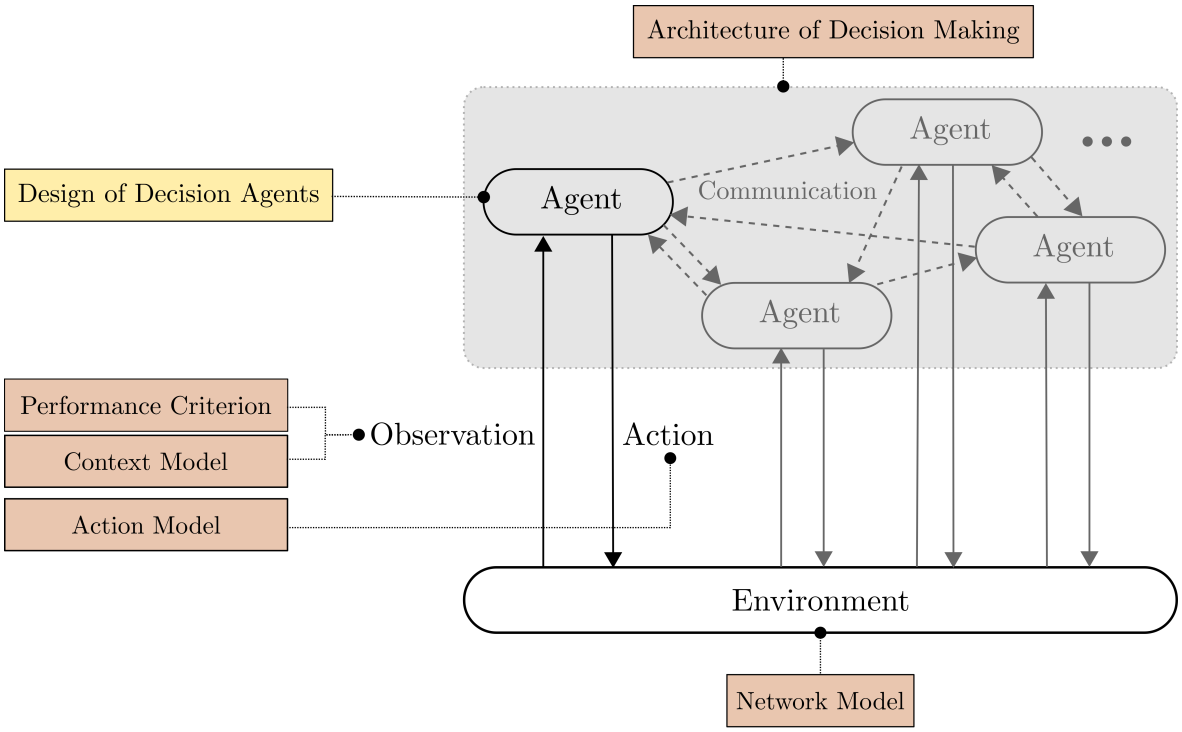


Figure 2.1. General agent-environment interaction and overview of context-aware decision making. Five components of system model for context-aware decision making shown in brown and design of decision agents shown in yellow.

observations [KAC+15] and interacting with each other via communication. Moreover, Figure 2.1 depicts how the five components of the system model for context-aware decision making in wireless networks, shown in brown, and the design of decision agents, shown in yellow, relate to the general agent-environment interaction. These relations will be explained in the following sections.

In Section 2.2, we introduce the considered system model for context-aware decision making in wireless networks, by first giving a brief overview of its five components in Section 2.2.1, and then discussing in more detail two of the components, the context model in Section 2.2.2 and the architecture of decision making in Section 2.2.3. Finally, in Section 2.3, we discuss the design a decision agents by pointing out different methods for decision making. In this regard, we also give a short introduction to two specific types of decision-making methods which are relevant for the remainder of this thesis, namely, optimization-based approaches and machine-learning-based approaches using multi-armed bandit frameworks.

2.2 System Model

2.2.1 Overview of Components

Modeling a context-aware decision-making framework for a new technique that jointly considers and leverages different types of resources in wireless networks, requires to formulate a system model, which typically consists of the following five components:

(i) A *model of the underlying wireless network*, including its different available *resources* and their *constraints*, is needed. Since the wireless network is part of the environment with which the decision agents in wireless networks interact, in Figure 2.1, the network model is hence related to the *environment* of the decision agents.

(ii) A *context model* is required that represents available side information that may be taken into account for decision making [Hen03, PZCG14, MSS13]. Such side information is observed by the decision agents, and in Figure 2.1 hence relates to the *observation* of the environment.

(iii) A *performance criterion* needs to be selected, with respect to which the performance of the system should be optimized. Moreover, it needs to be determined how a decision agent may *evaluate* the performance of its actions under a given situation with respect to the performance criterion, e.g., by evaluating a *utility function* or by receiving a *reward* from its environment [KAC+15]. Hence, the performance criterion and its evaluation also relates to the *observation* of the environment in Figure 2.1.

(iv) One or several adequate decision agents responsible for decision making need to be identified and an appropriate *architecture of decision making* needs to be designed within which the decision agents interact with each other and with the environment [Lun92, KB97, FCGS02]. Hence, in Figure 2.1, the architecture of decision making relates to the *set of decision agents and their interactions*.

(v) An *action model* needs to be defined, determining the different options that the decision agents may select within the technique (e.g., which resources to use in which way, or to allocate them to whom in which way). In Figure 2.1, the action model is hence related to the *action* taken by a decision agent.

Components (i), (iii) and (v) are common features of system models for resource allocation problems in wireless networks. In the following, we focus on components (ii) and (iv), by discussing in detail the context model and the architecture of decision making.

2.2.2 Context Model

The concept of *context awareness*, today perceived as important property of techniques for next generation wireless networks [BWL18], was first introduced in computer science in the area of *pervasive computing*, where it refers to computing systems which are able to acquire information about their environment and react based on changes in the environment [Sch95]. Since then, context awareness and context-aware computing have become important features in areas such as *context-aware communication* [SHT02], *wireless mobile autonomic computing and communications* [CFLP16] and *mobile and wireless networking* [MSS13] and are expected to also play an important role for the upcoming IoT paradigm [PZCG14].

Many definitions of context and context awareness have emerged over the years. An early comprehensive definition of *context* for context-aware applications was given in [DA99] according to which "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*" Moreover, in [DA99], *location*, *identity*, *activity*, and *time* are characterized to be the four primary types of context. The term *context awareness* has been defined in [DA99] as follows: "*A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.*" While we follow here [DA99], it should be noted that over the years, several other definitions of context and context awareness have been proposed [Hen03, MSS13, PZCG14].

Making use of context requires to formulate a *context model*. We follow here [Hen03], according to which "*A context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time.*" Using this definition from [Hen03], it becomes obvious that it is the duty of the designer to specify the required context. Moreover, this definition of a context model also reflects another requirement to achieve context-awareness, namely, that in

order to exploit context, such context first needs to be *acquired*. Hence, the designer needs to plan from which sources context should be acquired and what are feasible amounts of context to be acquired.

Given the above definitions, the *side information* which may affect the outcome of decision making for resource allocation in wireless networks, such as current network conditions and node-related, user-related or externally given conditions [BWL18,FSK+18], may conceptually be summarized under the term *context*. Hence, decision agents should take available context information into account for decision making. In wireless networks, context may be collected using *data collection resources* from several *sources* [MSS13], for instance, using the monitoring capabilities of base stations or of sensor-rich mobile devices. After context collection, data processing may be needed in order to aggregate and interpret the collected context and to produce context information which can be used by the decision agents [MSS13, BWL18]. Taking into account *context*, decision making becomes *context-aware*. In wireless communications, *context-aware decision making* has been studied before, for instance, for vertical handover decisions [ZJZ10, AKL06, FSM+15, FMS+17] and protocol reconfiguration in mobile devices [PAM10], see also [MSS13] for an overview.

2.2.3 Architecture of Decision Making

When designing a context-aware decision-making framework for a new technique that jointly considers and exploits different resources, the designer needs to select an appropriate *architecture of decision making* and to identify one or several adequate decision agents responsible for decision making. In this thesis, following [Lun92,KB97,FCGS02], we distinguish between the following architectures of decision making:

(i) *Centralized architecture*

(ii) *Decentralized architecture*

(iii) *Hierarchical architecture*

Which type of architecture is suitable highly depends on the specific technique, the specific performance criterion and the information required for decision making, such as network conditions and other relevant context. The properties of the three types of architectures are summarized in Table 2.1 and will be discussed in the sequel.

Table 2.1. Different types of architectures of decision making.

Type of architecture	**Decision agent(s)**	**Information collection**
Centralized	Central entity	At central entity
Decentralized	Local entities	At local entities
Hierarchical	Entities at multiple levels	At entities at multiple levels

First, in a *centralized* architecture, a central entity acts as global decision agent based on centrally collected information [Lun92]. This architecture may, for example, be useful if a network wide quantity needs to be optimized, i.e., a global performance criterion exists, and if it is feasible (e.g., in terms of communication overhead or privacy) that all needed information is collected at a central entity that selects suitable actions for individual nodes and informs the nodes about its decisions. While using centralized architectures facilitates solving global optimization problems up to optimality [FCGS02], they suffer from the following drawbacks. One disadvantage is that the central entity is a single point of failure, i.e., in the worst case, the operation of the whole network may break down if the central entity fails [KB97]. Moreover, a centralized architecture does not scale easily with increasing network size [KB97]. Classical examples of a centralized architecture of decision making are cellular networks, in which a base station acts as global decision agent, for example, taking care of user scheduling so as to avoid interference [Gol05].

Secondly, in a *decentralized* architecture, there are several local entities, each of them acting as local decision agents based on locally collected information [Lun92]. A decentralized architecture is, for instance, useful if either the nodes in the network have their own local performance criteria and a centralized architecture is not needed (e.g., since decisions of local entities do not affect each other) or if there exists a global performance criterion, but it is not desired or even feasible (e.g., in terms of communication overhead) to implement a centralized architecture [Lun92]. Decentralized architectures increase the reliability and robustness since no single point of failure exists and moreover, they are scalable [KB97]. However, in decentralized architectures, it is often difficult to reach globally optimal performance for global optimization problems [FCGS02]. In computing, decentralized architectures for decision making can, for example, be found in resource allocation for virtual machines in cloud computing [MF14, MMF17]. In wireless networking, they can, for instance, often be found in mobile ad-hoc networks [RT99, JHF03] and wireless sensor networks [SWKC12], where the wireless nodes themselves take local decisions.

Thirdly, it is possible to design the architecture *hierarchically*, i.e., in such a way that

information collection and decision making is split up between different decision agents at multiple hierarchically organized *levels*, enabled by an information exchange between them [KB97, FCGS02]. For instance, a set of local entities may be responsible for collecting the information required for decision making, while a central entity may be responsible for global decision making. By designing a suitable information processing at the local entities and a suitable information exchange between the local entities and the central entity, it is in certain cases possible to reduce the amount of data (compared to a centralized architecture) which needs to be transmitted to the central entity in order to enable the central entity to take globally optimal decisions [KTvK18]. As another example, decision making may be split up between multiple levels of decision agents, e.g., the decision agents at the higher levels take decisions which are used to coordinate the decision agents at the lower levels, while the decision agents at the lower levels take local decisions [Lun92]. Hierarchical architectures are, for example, useful if a network wide quantity needs to be optimized, i.e., a global performance criterion exists, but it is not feasible to share all needed information with a central entity. Hierarchical architectures try to combine the advantages of centralized and decentralized approaches, by being scalable [KB97], reliable [Lun92] and at the same time by facilitating to solve global optimization problems [FCGS02]. Nevertheless, it should be noted that using such a hierarchical architecture for solving a global optimization problem typically requires to decompose the global optimization problem into a hierarchical structure, which can be challenging [FCGS02]. Hierarchical architectures for decision making, for instance, can be found in resource allocation for virtual machines in cloud computing [MF11], they are also part of *fog computing* [BMZA12] and they are currently discussed in connection with fronthaul-constrained *cloud radio access network* (C-RAN) architectures [PWLP15, BASK18].

2.3 Design of Decision Agents

2.3.1 Methods for Decision Making

There are many different approaches how to design the decision agents for a new technique that jointly considers and exploits different resources in wireless networks. The design essentially depends on how the designer models the problem to be solved by the decision agent. Specifically, different approaches differ regarding how much of the decision making process is specified a priori by the designer and how the remaining problem is solved by the decision agent [KAC+15]. As outlined in [KAC+15], some important approaches to design decision agents are the following:

(i) *Explicit programming*

(ii) *Supervised learning*

(iii) *Optimization*

(iv) *Reinforcement learning* (RL)

In *explicit programming*, the designer explicitly programs the decisions to be taken by the decision agent for all possible situations it might face [KAC+15]. Since this approach may be infeasible for complex problems and for new techniques for wireless networks, we do not consider it for context-aware decision making in wireless networks.

Using *supervised learning*, a type of machine learning, the designer gives a set of training examples to the decision agent and the task of the decision agent is to learn the mapping from the input to the output [Alp14]. The decision agent runs a supervised learning algorithm in order to generalize from the training set. This approach requires expert knowledge from the designer who must build the training set of exemplary situations and the corresponding best actions [KAC+15]. Since such expert knowledge may not be available for new techniques for wireless networks, we do not consider it for context-aware decision making in wireless networks.

In *optimization*, the decision agent is given a set of actions, a set of constraints on these actions and a utility function to be maximized [BV04]. The utility function formalizes the performance criterion selected by the designer as a function of taking an action under a given situation. The decision agent may evaluate the performance of selecting an action by inserting it into the utility function. In order to search for optimal actions, the decision agent runs an optimization algorithm [KAC+15]. Hence, a designer may use such an *optimization-based approach* if the problem at hand can be modeled as an optimization problem whose parameters are known by the decision agent. Specifically, the designer needs to be able to define a utility function that formalizes the desired performance criterion as a function of taking an action under a given situation. Many problems in wireless communications can be modeled as mathematical optimization problems [SNHH15].

Using *reinforcement learning* (RL), a type of machine learning, the decision agent is given a set of actions, a set of constraints on these actions, and a performance criterion. The task of the decision agent is to find an action sequence which maximizes the performance criterion, but the performance of different actions in different situations is not known a priori [Alp14]. The agent has to test the different actions in order to

discover which performance, or in RL language, *reward*, each action yields [SB98]. For this purpose, the decision agent runs an online learning algorithm that sequentially selects actions in order to learn the performances of different actions under different situations and thereby maximize the cumulative performance over time. Hence, applying a *machine-learning-based approach* using RL makes sense if the designer can (or would like to) only provide a performance criterion to the decision agent and wants the decision agent to learn how to take actions by interacting with an uncertain environment [SB98]. Special cases of RL are *multi-armed bandit* (MAB) frameworks [Rob52, ACBF02]. Here, the choice of action only impacts the immediate outcome, but not the outcomes of future action selections. MAB frameworks have become a useful tool to tackle problems in wireless communications [MH16, JZR+17].

In the sequel, we will give a short overview of optimization and MABs since these types of problems and the corresponding solution methods are highly relevant for context-aware decision making in wireless networks and will be used in the remainder of this thesis.

2.3.2 Optimization

2.3.2.1 General Problem Formulation

Many problems appearing in wireless communications and, in particular, many resource allocation problems can be modeled as mathematical optimization problems [SNHH15]. A (mathematical) *optimization problem* can be formulated as

$$\begin{aligned} &\min\ g_0(\mathbf{y}) \\ &\text{s.t. } g_j(\mathbf{y}) \leq 0,\ j = 1, ..., J, \\ &\quad \mathbf{y} \in \mathbb{R}^k, \end{aligned} \tag{2.1}$$

where $\mathbf{y} \in \mathbb{R}^k$ is the vector of *optimization variables*, the function $g_0 : \mathbb{R}^k \to \mathbb{R}$ is the *objective function* and the functions $g_j : \mathbb{R}^k \to \mathbb{R}$, $j = 1, ..., J$, are the *constraint functions* [BV04]. Moreover, the abbreviations "min" and "s.t." stand for *minimize* and *subject to*, respectively.

2.3.2.2 Classes of Optimization Problems

Different classes of optimization problems are distinguished according to the type of objective and constraint functions. If the objective function g_0 and the constraint

functions g_j, $j = 1, ..., J$, are linear functions of the optimization variables $\mathbf{y}$, then Problem (2.1) is called a *linear programming problem*. In this case, any locally optimal point is also globally optimal [BV04]. While no analytical formula for the solution of the general linear programming problem exists, effective methods exist that can find the optimal solution reliably and efficiently [BV04]. More precisely, linear programming is *solvable in polynomial time* [Kar84].

If the objective function or the constraint functions are nonlinear, Problem (2.1) is called a *nonlinear programming problem*. Nonlinear programming problems may, in general, have several local optima [SNHH15] and there are no effective methods for finding the optimal solution of the general nonlinear programming problem [BV04].

A special case of a nonlinear programming problem is a *convex optimization problem*, in which the objective and constraint functions are convex. One characteristic of convex optimization problems is that any locally optimal point is also globally optimal [BV04]. While no analytical formula for the solution of the general convex optimization problem exists, effective methods exist that can solve even large instances of convex problems reliably and efficiently [BV04]. In particular, many classes of convex optimization problems can be solved in polynomial time [NN94].

If the objective and constraint functions in Problem (2.1) are linear and the optimization variables are restricted to be integers, Problem (2.1) is called *integer linear programming* (ILP) problem. The general ILP problem belongs to the complexity class of *non-deterministic polynomial-time* (NP-hard) optimization problems [Sch86]. There exists *no polynomial-time algorithm* for the solution of NP-hard optimization problems – unless P=NP, where P is the class of problems which can be solved in polynomial time, which would imply that all NP-hard optimization problems could be solved by a polynomial-time algorithm [CLRS09]. It is widely believed that P $\neq$ NP and that hence NP-hard optimization problems are not solvable by any algorithm whose running time is polynomially bounded in the size of the input for the algorithm [CLRS09]. Therefore, the general ILP problem is believed not to be solvable efficiently to optimality [Sch86]. Indeed, while the optimal solution could be found by enumerating all feasible solutions, such an approach in general results in an *exponential-time algorithm*, i.e., an algorithm which can only be exponentially bounded in the size of the input for the algorithm [Sch86]. A special case of an ILP problem is the *knapsack problem*, which will be discussed next since it is needed in the remainder of this thesis.

2.3.2.3 The Knapsack Problem

The *knapsack problem* is a well-known optimization problem and a special case of integer programming [KPP04]. Formally, it may be described as follows. A set $\mathcal{I} = \{1, ..., I\}$ of items is given. Each item $i \in \mathcal{I}$ is associated with a *profit* p_i and a *weight* w_i. Moreover, there is a *capacity value* c. The goal is to select a subset of items which maximizes the sum profit of the selected items, while the sum weight of the selected items may not exceed the capacity value c. The knapsack problem can be formulated as an ILP problem with binary variables as follows:

$$\begin{aligned} \max \ & \sum_{i=1}^{I} p_i y_i \\ \text{s.t.} \ & \sum_{i=1}^{I} w_i y_i \leq c \\ & y_i \in \{0, 1\} \text{ for } i = 1, ..., I. \end{aligned} \tag{2.2}$$

Without loss of generality, the parameters appearing in Problem (2.2) may be assumed to satisfy $p_i > 0$, $w_i > 0$ for all $i \in \mathcal{I}$ and $c > 0$. This is because variables corresponding to non-positive parameters could immediately be fixed to their optimal values as a pre-processing step, whereby the overall problem is transformed into an equivalent problem which then satisfies the above assumptions [KPP04].

The knapsack problem is known to belong to the class of NP-hard optimization problems [KPP04]. Among the approaches to solve the knapsack problem are methods known more generally from integer programming, such as *dynamic programming* and *branch-and-bound*, which may be used to solve the knapsack problem up to optimality, but can be extremely time and memory consuming [KPP04]. Other approaches for the knapsack problem are so-called *approximation algorithms* and *approximation schemes* that may find "good enough" solutions in a "reasonable" amount of time [KPP04]. Finally, many heuristics exist, one of them being the so-called *greedy algorithm for the knapsack problem* [KPP04]. The idea of this greedy algorithm is as follows. For each item i, the *profit to weight ratio*, or *efficiency*, defined by $\text{eff}_i := \frac{p_i}{w_i}$ is computed. Then, items are sorted in decreasing order with respect to their efficiency. Finally, as long as sufficient capacity is left, items are added to the knapsack one after the other according to the ranking by starting with the items of highest efficiency. While the solution of the greedy algorithm may, in general, be arbitrarily bad on certain instances of the knapsack problem, it is possible to extend the algorithm in a simple fashion to yield an approximation algorithm for the knapsack problem, whose profit is guaranteed to be at least half of that of the optimal solution [KPP04].

The knapsack problem is important since it may be seen as the simplest ILP problem, which itself appears as a sub-problem in many more complex optimization problems [SNHH15]. Moreover, the knapsack problem and its many variants have a large number of applications [KPP04]. In wireless communications, different variants of knapsack problems have, for instance, occurred in opportunistic scheduling [LK03], coded caching for wireless content delivery [PT13] and resource allocation in OFDM wireless networks [KNY09]. A particular variant of the knapsack problem relevant for the remainder of this thesis is the *multi-dimensional knapsack problem*, which will be discussed next.

2.3.2.4 The Multi-Dimensional Knapsack Problem

The *multi-dimensional knapsack problem* is a generalization of the basic knapsack problem, where instead of one capacity constraint, d different capacity constraints occur, for an integer $d \geq 2$ [KPP04, Fré04]. Formally, again a set $\mathcal{I} = \{1, ..., I\}$ of items is given. Each item $i \in \mathcal{I}$ is associated with a profit p_i and with a weight $w_{i,j}$ with respect to d different attributes $j = 1, ..., d$. Moreover, there is a capacity value c_j for each attribute $j = 1, ..., d$. The goal is to select a subset of items which maximizes the sum profit of the selected items, while the sum weight of the selected items with respect to any of the attributes may not exceed the corresponding capacity value. The multi-dimensional knapsack problem can be formulated as an ILP problem with binary variables as follows:

$$\begin{aligned} \max \ & \sum_{i=1}^{I} p_i y_i \\ \text{s.t.} \ & \sum_{i=1}^{I} w_{i,j} y_i \leq c_j \text{ for } j = 1, ..., d. \\ & y_i \in \{0, 1\} \text{ for } i = 1, ..., I. \end{aligned} \tag{2.3}$$

The parameters appearing in Problem (2.3) may be assumed to satisfy $p_i > 0$, $w_{i,j} \geq 0$ for all $i \in \mathcal{I}$, $j = 1, ..., d$, and $c_j > 0$ for all $j = 1, ..., d$. Like for the basic knapsack problem, this is because variables corresponding to parameters which do not satisfy these assumptions could immediately be fixed to their optimal values as a pre-processing step [KPP04, Fré04]. Thereby, the overall problem is transformed into an equivalent problem which then satisfies the above assumptions.

Like the knapsack problem, the multi-dimensional knapsack problem belongs to the class of NP-hard optimization problems [KPP04, Fré04]. Approaches to solve the

multi-dimensional knapsack problem up to optimality include *branch-and-bound algorithms* [GP85] and *dynamic programming* [HGI18]. In general, such exact approaches are often extremely time and memory consuming for large problem instances [KPP04]. Moreover, it has been shown that the multiple constraints make the multi-dimensional knapsack problem much more difficult compared to the basic one-dimensional knapsack problem. Specifically, there exists no *fully polynomial time approximation scheme* for the multi-dimensional knapsack problem, unless P=NP, meaning that no efficient approximation of this problem is believed to be possible [KPP04, Fré04]. Due to the difficulty of the multi-dimensional knapsack problem, many heuristics exist, which usually cannot give performance guarantees, such as, greedy-type heuristics inspired by the greedy algorithm for the one-dimensional knapsack problem [Dob82]. In the design of greedy-type heuristics for the multi-dimensional knapsack problem, there exist different approaches how to define the efficiency of an item [KPP04].

2.3.3 Multi-Armed Bandits

2.3.3.1 Balancing Exploration and Exploitation

Multi-armed bandit (MAB) problems are problems of sequential decision making under uncertainty and constitute a special case of RL [SB98]. The term *multi-armed bandit* is referring to the sequential allocation problem faced by a gambler in a casino who faces several slot machines (i.e., *bandits*) and needs to sequentially select one slot machine to play (i.e., the gambler needs to sequentially decide *which arm to pull*) [BC12]. In the most basic MAB formulation, given a set of *actions* (also called *arms*), an agent selects one action per round and receives a *reward* which depends on the selected action [Rob52, ACBF02]. The goal of the agent is to *maximize its cumulative reward* over a sequence of rounds up to the time horizon. However, the agent does not know the reward distributions of the actions. Instead, it may only observe instantaneous rewards of selected actions. Selecting a suboptimal action may lead to a loss in terms of reward. This loss is called the *regret* of learning and the agent hence tries to minimize the total regret over the time horizon [ACBF02].

In order to learn about the rewards of the actions, the agent needs to try out the different actions over time. On the one hand, the agent needs to explore actions, about which it has little information available. *Exploration* may lead to low rewards in the short run, but it is needed in order to learn about the rewards of all actions and thereby receive higher rewards in the long run [SB98]. On the other hand, the agent needs to exploit knowledge obtained so far, by selecting those actions which it already

discovered to yield high rewards. *Exploitation* helps the agent to immediately receive high rewards [SB98]. Finding a good *trade-off between exploration and exploitation* is crucial in order to yield high cumulative rewards over the time horizon [SB98].

A solution to a MAB problem is a *policy* or *allocation strategy* according to which the actions should be selected throughout the sequence of rounds [ACBF02]. Such a policy is given by a *learning algorithm*, which maps the history of action selections and the associated obtained rewards to the next action selection. The performance of a learning algorithm may be evaluated in terms of its regret, i.e., the difference in reward which could have been achieved had the agent selected the best action and the reward that the agent actually achieved [ACBF02]. If the total expected regret $R(T)$ of a learning algorithm after T rounds satisfies $\lim_{T\to\infty} \frac{R(T)}{T} = 0$, then the learning algorithm converges to the optimal allocation strategy when the number T of rounds goes to infinity. Specifically, this is the case if $R(T)$ is a sublinear function in T, i.e., if there exists $\gamma < 1$, such that $R(T) = O(T^\gamma)$ [TvdS15a]. Here, $O(\cdot)$ denotes the O-notation, which can be used to characterize the limiting behavior of a function by giving an *asymptotic upper bound* on its growth rate, see [CLRS09] for a formal definition.

Various problems in wireless communications have been posed as MAB problems [MH16], such as cognitive jamming [ATvdSB16] or mobility management [STv16]. Modeling problems occurring in wireless communication networks via MABs is appealing since MAB models naturally comprise missing a priori knowledge and limited feedback, both of which are typical challenges in wireless communication networks where information exchange may be costly [MH16]. Moreover, many features of wireless networks may actually be covered using MAB models since, apart from the most basic MAB problem discussed above, a large variety of further MAB models exist, which will be briefly discussed next.

2.3.3.2 Types of Multi-Armed Bandit Models

Among the variety of MAB models, for instance, the following features are covered. Different MAB models make different assumptions on the nature of the reward process. While the reward processes are assumed to be *stochastic* in the basic model described above, i.e., the model is stateless and the rewards are generated according to a specific state-independent probability distribution, other MAB models assume that the reward processes are *adversarial*, and again other MAB models assume that the reward processes are *Markovian*, i.e., each action has a state that changes over time according to a Markov process [BC12, MH16].

In MAB models with *dependent arms*, dependencies between the rewards of different actions are taken into account [PCA07]. In *combinatorial* MAB models, several actions may be taken per round and the reward may depend on the subset of selected actions [CWY13]. In *contextual* MAB models, in each round, the agent receives side information that may affect the rewards of the actions [BC12]. In *sleeping* MAB models, the set of actions may vary over time and the availability of actions may evolve in a stochastic, Markovian or adversarial manner [KNMS10]. In *multi-agent* MAB models, several agents exist, each one of them aiming at maximizing its reward over its own set of actions, where the reward received by an agent may depend on the selected actions of the other agents [MH16].

Next, since it is relevant for this thesis, we will formally introduce the most basic MAB problem and discuss learning algorithms proposed for this problem.

2.3.3.3 The Stochastic Multi-Armed Bandit Problem

Formally, the most basic MAB problem, called *stochastic multi-armed bandit problem*, is given as follows [Rob52, ACBF02]. Consider a set $\mathcal{A}$ of $A := |\mathcal{A}|$ actions and a time horizon T. Over a sequence of rounds $t = 1, ..., T$, an agent may select one action per round. Playing an action $a \in \mathcal{A}$ yields a reward sampled from an unknown reward distribution with expected value denoted by μ_a. Rewards are independent across different actions. Moreover, rewards for the same action are independent and identically distributed across different rounds.

Let a^* denote the optimal action in expectation, i.e., $a^* = \text{argmax}_{a \in \mathcal{A}} \mu_a$. An *oracle* with a priori knowledge about the expected values $\{\mu_a\}_{a \in \mathcal{A}}$ would select a^* in each of the T rounds, yielding an expected cumulative reward of $T \cdot \mu_{a^*}$ [ACBF02]. In contrast, a learning algorithm that selects action $a_t \in \mathcal{A}$ in round t for $t = 1, ..., T$, yields an expected cumulative reward of $\mathbb{E}[\sum_{t=1}^{T} \mu_{a_t}] = \sum_{a \in \mathcal{A}} \mu_a \, \mathbb{E}[N_a(T)]$, where $N_a(T)$ denotes the number of times action $a \in \mathcal{A}$ has been played over the rounds $t = 1, ..., T$, and where the expectation is taken with respect to the action selections by the learning algorithm and the randomness of the reward distributions [ACBF02]. The regret $R(T)$ of the learning algorithm with respect to the oracle after T rounds is defined as

$$R(T) := T \cdot \mu_{a^*} - \sum_{a \in \mathcal{A}} \mu_a \, \mathbb{E}[N_a(T)]. \tag{2.4}$$

A simple heuristic policy for the stochastic multi-armed bandit problem is the *ϵ-greedy algorithm* [SB98], which works as follows. First, a parameter $\epsilon \in (0, 1)$ is selected.

Then, in each round $t = 1, ..., T$, with a probability of $(1 - \epsilon)$, the ϵ-greedy algorithm selects the action with the highest empirical mean reward based on the history of previous selections and obtained rewards. With a probability of ϵ, the ϵ-greedy algorithm selects a random action. For a constant ϵ, the regret $R(T)$ of the ϵ-greedy algorithm can only be bounded linearly in the horizon T [KP00]. Therefore, the algorithm does in general not converge to the optimal allocation strategy. If the algorithm is adapted by decreasing ϵ in a certain fashion over time, logarithmic regret bounds may be achieved [ACBF02]. However, as in [KP00], in our numerical simulations, we only consider the standard ϵ-greedy algorithm with constant ϵ since its variants were not found to be beneficial in practice [VM05].

A well-known learning algorithm for the stochastic multi-armed bandit problem is the *UCB1 algorithm* [ACBF02]. UCB1 is based on the following idea. For each action, an *upper confidence bound* (UCB) of the expected reward of each action is estimated. Then, in each round, the action with the highest estimated upper confidence bound is selected. The UCB1 policy achieves logarithmic regret uniformly over T for any reward distributions of known bounded support [ACBF02]. Hence, $R(T) = O(\log(T))$ holds. Moreover, it has been shown that the regret of the stochastic multi-armed bandit problem for any policy is growing at least logarithmically, i.e., $R(T) = \Omega(\log(T))$ [LR85]. Here, $\Omega(\cdot)$ denotes the Ω-notation, which can be used to characterize the limiting behavior of a function by giving an *asymptotic lower bound* on its growth rate, see [CLRS09] for a formal definition. Hence, UCB1 achieves the optimal regret up to a multiplicative constant. Different variants of UCB-type policies have been proposed over time also for other variants of the MAB problem. For example, the *awake upper estimated reward* (AUER) algorithm proposed in [KNMS10] is an extension of UCB1 to the sleeping arm case, where the set of available actions is assumed to vary over time according to the selections made by an adversary. Moreover, the LinUCB algorithm in [LCLS10, CLRS11] uses a UCB-type approach for a contextual MAB problem with linear payoff functions.

Since contextual MAB problems are important in the remainder of this thesis, they will be discussed next.

2.3.3.4 Contextual Multi-Armed Bandit Problems

In *contextual MAB problems*, also called *bandits with side information* [BC12], in each round, the agent first observes a side information, called *context*, before choosing among a set of actions with unknown rewards. The reward received from playing an action may depend on the observed context. The agent hence needs to learn which action has the

highest reward in which context, i.e., the best mapping from contexts to actions [BC12], in order to maximize its expected reward over time.

While there exist various models for contextual MABs, we next present a specific contextual MAB model which is based on the models presented in [TZvdS14, TvdS15a]. The model presented next will be useful in the remainder of this thesis. Consider a set $\mathcal{A}$ of $A := |\mathcal{A}|$ actions, a bounded context space $\mathcal{X} := [0,1]^D$ of D dimensions, and a finite time horizon T. In each round $t = 1, ..., T$, the following events happen sequentially:

(i) A context $\mathbf{x}_t \in \mathcal{X}$ is revealed to the agent.

(ii) The agent selects an action $a_t \in \mathcal{A}$.

(iii) The agent receives a reward r_{t,a_t}.

The reward r_{t,a_t} is sampled from an unknown reward distribution which depends on the selected action a_t and on the context $\mathbf{x}_t$. The expected value of this reward distribution of action a_t under context $\mathbf{x}_t$ is denoted by $\mu_{a_t}(\mathbf{x}_t)$. Hence, $\mathbb{E}[r_{t,a_t}] = \mu_{a_t}(\mathbf{x}_t)$ holds. Rewards are independent across different actions, and rewards for the same action are independent across different rounds. Moreover, the sequence $\{\mathbf{x}_t\}_{t=1,...,T}$ of context arrivals is generated before the first round, i.e., the actions selected by the agent do not influence the context arrivals. Finally, it is assumed that a similarity metric over the context space holds. Specifically, the similarity metric is given by the following *Hölder continuity assumption*. There exist $L > 0$ and $0 < \alpha \leq 1$ such that

$$|\mu_a(\mathbf{x}) - \mu_a(\tilde{\mathbf{x}})| \leq L||\mathbf{x} - \tilde{\mathbf{x}}||_D^\alpha \tag{2.5}$$

holds for all $a \in \mathcal{A}$ and for all $\mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X}$, where $||\cdot||_D$ denotes the Euclidean norm in $\mathbb{R}^D$. The Hölder continuity assumption reflects that the expected reward of an action is similar in similar contexts.

Let $a^*(\mathbf{x}_t)$ denote the optimal action with respect to context $\mathbf{x}_t$ in expectation, i.e., $a^*(\mathbf{x}_t) = \text{argmax}_{a \in \mathcal{A}} \mu_a(\mathbf{x}_t)$. An oracle with a priori knowledge about the expected values $\{\mu_a(\mathbf{x})\}_{a \in \mathcal{A}, \mathbf{x} \in \mathcal{X}}$ would select $a^*(\mathbf{x}_t)$ in round t, $t = 1, ..., T$, yielding an expected cumulative reward of $\mathbb{E}\left[\sum_{t=1}^T r_{t,a^*(\mathbf{x}_t)}\right] = \sum_{t=1}^T \mu_{a^*(\mathbf{x}_t)}$ [TvdS15a]. In contrast, a learning algorithm that selects action $a_t \in \mathcal{A}$ in round t for $t = 1, ..., T$, yields an expected cumulative reward of $\mathbb{E}\left[\sum_{t=1}^T r_{t,a_t}\right]$, where the expectation is taken with respect to the action selections by the learning algorithm and the randomness of the reward distributions [TvdS15a]. Given an arbitrary sequence $\{\mathbf{x}_t\}_{t=1,...,T}$ of context

arrivals, the regret $R(T)$ of the learning algorithm with respect to the oracle after T rounds is defined as

$$R(T) := \sum_{t=1}^{T} \mu_{a^*(\mathbf{x}_t)} - \mathbb{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right]. \tag{2.6}$$

Various other models for contextual MABs exist. While all of them have in common that side information is revealed to the agent, these models differ regarding their assumptions on how context is generated and on how rewards are formed. Context arrivals may be assumed to be stochastic, e.g., *independent identically distributed* (i.i.d.) or non-stationary, or of adversarial nature [BC12]. Moreover, more or less strong assumptions on how rewards are formed may be made. For example, a linear dependency between reward and context [LCLS10] or a known similarity metric over the context space [Sli14] may be assumed. Depending on the specific underlying contextual MAB model, algorithms are proposed tailored to these models and then typically shown to achieve sublinear upper regret bounds.

Some algorithms are based on contextual MAB models like the one presented above, which assumes a known similarity metric over the contexts. Algorithms for this type of contextual MAB model typically group contexts into sets of similar contexts by partitioning the context space. Then, they estimate the reward of an action under a given context based on previous rewards for that action in the set of similar contexts. For example, based on the model presented above, the algorithms proposed in [TvdS15a, TZvdS14] use uniform and non-uniform adaptive partitions of the context space. Moreover, for a model where a *Lipschitz similarity metric* between context-action pairs is assumed, [LPP10] proposes a uniform partition of the context space, and the *contextual zooming algorithm* [Sli14] proposes a non-uniform adaptive partition of the context space. The algorithms in [LPP10, Sli14, TvdS15a, TZvdS14] not only work for finite, but also for an infinite set of actions.

In contrast, other algorithms are based on models with stronger assumptions regarding how rewards are formed. For example, the *LinUCB algorithm* in [LCLS10, CLRS11] assumes that the expected reward is linear in the context. A linearity assumption is also used in the algorithm based on Thompson-sampling in [AG13], and in [GLZ14], where a clustering is performed on top of a contextual MAB setting.

Again other algorithms are based on models with fewer assumptions regarding how rewards are formed. For instance, the *epoch-greedy algorithm* relies on a general contextual MAB model with no further assumptions on how rewards are formed [LZ07].

Also the algorithm in [BLS14] for contextual MABs with resource constraints and policy sets makes no further assumptions on how rewards are formed, except that it is assumed that the marginal distribution over the contexts is known. However, compared to [Sli14, TvdS15a, TZvdS14], the algorithms in [LZ07, BLS14] work only for a finite set of actions and are based on the assumption that in each time step, the tuples (context, rewards) are sampled from a fixed, but unknown distribution, i.e., contexts are generated in an i.i.d. fashion. Therefore, while the algorithms in [LZ07, BLS14] are more general regarding how rewards are formed, they are more restrictive regarding the context arrivals since they require context arrivals to be i.i.d. over time.

Algorithms for contextual MAB problems may also be distinguished based on their approach to balance the exploration vs. exploitation trade-off. While the epoch-greedy algorithm [LZ07] and the algorithms in [TvdS15a, TZvdS14] explicitly distinguish between exploration and exploitation steps, the LinUCB algorithm [LCLS10, CLRS11], the contextual zooming algorithm [Sli14], and the algorithms in [GLZ14, LPP10] follow an index-based approach, in which in each round, the action with the highest index is selected. Other algorithms, like the one for contextual MABs with resource constraints in [BLS14], draw samples from a distribution to find a policy which is then used to select the action. Finally, the algorithm based on Thompson-sampling in [AG13] draws samples from a distribution to build a belief, and selects the action which maximizes the reward based on this belief.

Chapter 3

Computation Offloading in Wireless Multi-Hop Networks

3.1 Introduction

In this chapter, we consider computation offloading in multi-hop wireless networks, a technique that trades *communication resources* off against *computation resources*. Using computation offloading, wirelessly connected mobile devices may offload computation tasks to resource-rich servers for remote computation by transmitting the data required to remotely process the tasks to the servers [KLLB13]. Such servers may, for instance, be part of centralized computing platforms accessible via the Internet as in *mobile cloud computing* (MCC) [DLNW13] or they may be part of computing infrastructure placed at the edge of the wireless networks as in *mobile edge computing* (MEC) [HPS+15, WZZ+17, MYZ+17]. Computation offloading may improve the performance of wirelessly connected mobile devices by reducing task completion times, and it also may reduce the devices' energy consumption [KLLB13]. Thereby, the usage of resource-hungry applications in future mobile and IoT devices may be facilitated. The benefit of computation offloading for an individual device in terms of its battery life depends on whether less energy has to be spent for transmitting the task to the server than for local processing. Whether the latter is the case, in turn, depends not only on channel conditions and computing capabilities of the device, but also on task characteristics [KL10, MN10]. Therefore, when deciding whether or not to offload a task, context information should be taken into account. Computation offloading has so far not been considered in multi-hop networks, where network coverage may be extended and required transmission power reduced. As motivated in Section 1.3.2, in multi-hop networks, offloading decisions are non-trivially coupled since communication resources of relay nodes need to be used and shared for task offloading.

Hence, we here investigate the problem of context-aware computation offloading for energy minimization in multi-hop wireless networks. In this chapter, we propose an optimization-based approach and a centralized architecture of decision making. We use an optimization-based approach since the parameters appearing in the problem may be assumed to be known. Moreover, we use a centralized architecture of decision making, cf. Section 2.2.3, since the offloading decisions of the nodes are non-trivially

coupled and hence need to be jointly optimized. Specifically, we propose a context-aware greedy heuristic algorithm for computation offloading in multi-hop networks. Using this algorithm, a central entity may take offloading decisions based on centrally collected information about network conditions and task context. This chapter presents work originally published by the author in [MASW+15]. Compared to [MASW+15], in this thesis, we additionally study the computational complexity of the proposed algorithm and its overhead by analyzing the communication requirements under the proposed centralized architecture of decision making. In addition, in this thesis, we additionally highlight the ideas of the mathematical proofs in the main body of text, while the full mathematical proofs are given in the appendices. Moreover, in this thesis, the numerical simulations are extended.

The remainder of this chapter is organized as follows. In Section 3.2, we give a detailed review of the state of the art on decision making for computation offloading. In Section 3.3, the system model for context-aware computation offloading in multi-hop networks is introduced. Section 3.4 provides a formal problem formulation of context-aware computation offloading for energy minimization in multi-hop wireless networks using an optimization-based approach and a centralized architecture of decision making. In Section 3.5, the optimization problem is analyzed. In Section 3.6, a context-aware greedy heuristic algorithm for computation offloading in multi-hop networks is proposed. Section 3.7 discusses properties of the proposed algorithm. In Section 3.8, the performance of the proposed algorithm is numerically evaluated. Section 3.9 concludes this chapter.

3.2 State of the Art

In this section, a review of the state of the art on decision making for computation offloading is presented. This review complements the short review presented in Section 1.3.2 by discussing in detail the works introduced in Table 1.1.

Previous works have mainly considered computation offloading in single-hop networks where devices have a direct connection to a server whose computation resources they may utilize. One line of work considers computation offloading from the point of view of one single mobile device. These works aim at energy or completion time minimization by designing mechanisms to decide whether to offload and which parts of an application to offload. In [KL10] and [MN10], conditions are derived under which a mobile device may save energy by using computation offloading. The calculations

in [KL10] and [MN10] show that energy savings depend on the ratio of "communication vs. computation". This ratio not only depends on the communication conditions (e.g., allocated bandwidth) and computation capabilities of the mobile device (e.g., processor speed), but also on the specific characteristics of the tasks to be offloaded. Computation offloading is especially beneficial for applications with high computational requirements, but low amount of data to be transmitted. In [XLL07], a timeout scheme is presented which allows a mobile device to save energy by offloading parts of its computation. Their proposed approach is proven to be 2-competitive, i.e., the ratio between the performance of the proposed approach and the performance of an optimal offline algorithm can be bounded by a factor of 2. In [WZL12], an analytical solution is given for minimizing the consumed energy by optimal selection of both the clock-frequency in case of local computation and the data rate over time in case of computation offloading. In [HWN12], a dynamic offloading algorithm based on Lyapunov optimization is presented, which achieves energy savings for a mobile device by deciding which components of a software should be executed remotely under the current network conditions. In [RP03], a policy for energy-optimal remote processing in a client-server system based on Markov models is proposed which optimizes the energy consumption at the client. In [LMZL16], policies for task scheduling in a single device are derived based on a Markov decision process. The proposed algorithm minimizes the average delay of each task under a power constraint. In the survey on computation offloading for mobile systems in [KLLB13], conditions are derived, under which offloading is beneficial with respect to both the task completion time and the device's energy consumption.

Newer works consider the dynamics among several mobile devices using computation offloading in single-hop networks, e.g., concerning the traffic induced by computation offloading or the competition for shared resources. In [GZQL12], energy minimization of computation offloading in a single-hop network is investigated, where mobile devices may choose between several servers. Here, interdependencies between the devices' decisions arise from congestion at popular servers and a game-theoretic model is used to analyze the decentralized dynamics. In [MBASK18], a distributed game-theoretic algorithm for energy minimization in multi-stage computation offloading is proposed, where mobile devices in a single-hop network may either compute their tasks locally or offload it to either the AP or to a cloud server. The proposed algorithm iteratively takes care of both resource allocation and offloading decisions. In [NMAS+18], a game-theoretic framework is proposed for computation offloading in a single-hop network, where mobile users may offload fractions of their tasks. The proposed algorithm minimizes the completion time of the tasks in the network and is shown to reach the globally optimal solution. In [Che15], a decentralized game-theoretic framework for

computation offloading in a single-hop network is introduced which models both energy and time costs of each single device based on the offloading decisions of others. Using their mechanism, mobile devices take local offloading decisions that are beneficial for the overall system performance. In [CLD16], the offloading decisions and the resource allocation for multiple users in a single-hop network are jointly optimized with the goal to minimize both the energy and time costs in the network. The proposed heuristic algorithm based on separable semidefinite relaxation centrally computes both the offloading decisions as well as the resource allocation.

A multi-hop scenario has been considered in a different context of "communication vs. computation". In [MYM02] and [TF09], multi-media sensor networks are investigated, in which data may be compressed at sensor nodes before communicating it to a central entity in a multi-hop fashion.

However, to the best of our knowledge, computation offloading has not yet been considered in multi-hop networks, which may extend network coverage and reduce required transmission power. Decision-making for computation offloading is challenging in multi-hop networks as offloading decisions are interdependent due to the need to use and share communication resources of relay nodes, which may even have their own computation tasks.

3.3 System Model

3.3.1 Introduction

In this section, we propose a general model for context-aware computation offloading in multi-hop wireless networks. In accordance with Section 2.2.1, the proposed general model consists of the following five components:

(i) A *network model* is formulated to specify the assumptions on the multi-hop communication in the underlying multi-hop wireless network.

(ii) A *context model* is defined, which is used to characterize the devices' computation tasks.

(iii) As performance criterion to be minimized, the network energy consumption is considered and an *energy consumption model for communication and computation*

is formulated, which can be used to compute for each task the required energy for computing the task locally and the required energy for transmitting the task to the server.

(iv) A centralized *architecture of decision making* is proposed, where a central entity takes offloading decisions.

(v) An *action model* is determined, which determines the different choices of the central entity, namely, which tasks should be offloaded and which should not.

The proposed general model for context-aware computation offloading in multi-hop wireless networks is applicable to multi-hop networks of any topology in which an AP gives access to a resource-rich server. Moreover, the model is compatible with different types of infrastructures for computation offloading, such as in MCC and MEC.

3.3.2 Network Model

We consider an ad-hoc wireless multi-hop network consisting of a set $\mathcal{N} = \{1, ..., N\}$ of $N := |\mathcal{N}| \geq 2$ mobile devices, called *nodes* 1 to N. We assume there exists a *server* connected to a stable energy supply, which is capable of parallel task processing and which offers computing resources to the mobile devices as a service for computation offloading [SHP+14]. More specifically, mobile devices may rent virtual machines in the server with guaranteed properties (e.g., certain number of cores and clock rate) in order to offload their tasks [SHP+14]. Since we focus on the problem of which devices should offload their tasks due to the scarcity of communication and computation resources within the wireless multi-hop network, we assume here that the computation resources in the server are sufficient for all devices to offload their tasks. How to manage the computation resources in a cloud server with limited computation is, for instance, discussed in [MF11, MF14, MMF17]. Moreover, how to select among different types of virtual machines offered by a cloud server from mobile device perspective is, for example, discussed in [SHP+14].

We further assume that there exists an AP via which the server may be reached. Our model is generic with respect to the infrastructure of computation offloading, i.e., the server may either be directly attached to the AP, as in MEC [MYZ+17], or the server may be located in a distant cloud and reached via the AP over the Internet, as in MCC [DLNW13]. An illustration of the considered network model is depicted in Figure 3.1, where a multi-hop wireless network with an AP giving access to a server is

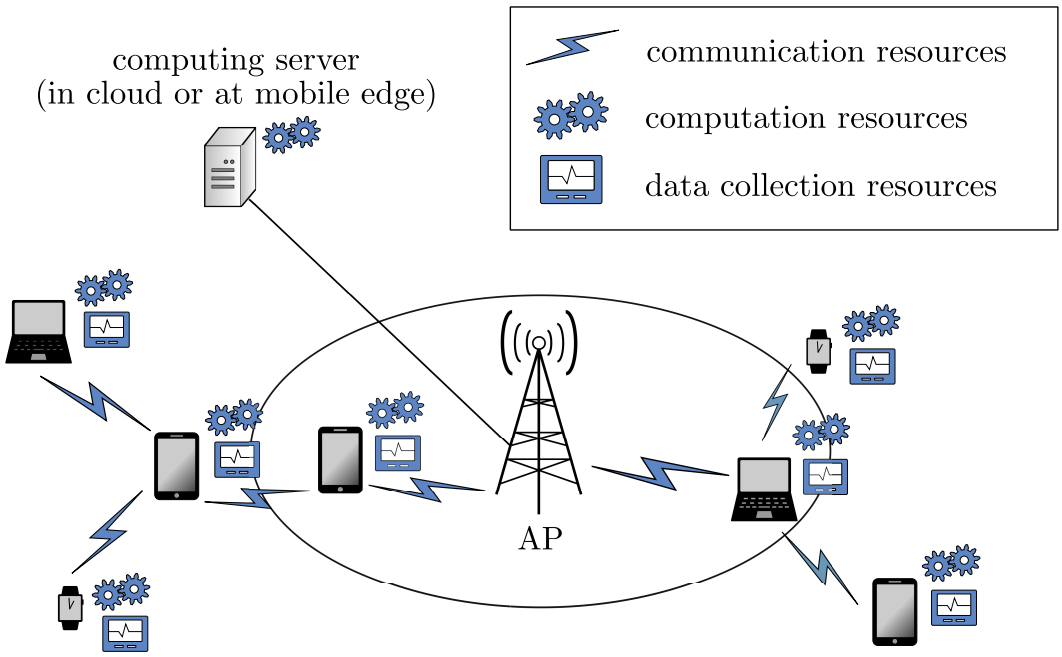


Figure 3.1. Network model.

shown. The nodes in the network dispose of communication, computation and data collection resources. The data collection resources are shown here since for taking meaningful offloading decisions, information about channel conditions, computing capabilities of the devices, and task characteristics needs to be collected, as will be explained in the next sections.

We consider a computation offloading session during which the location of the nodes – which may change in between different sessions – is assumed to be fixed. Such an assumption is reasonable for scenarios in which nodes move slowly compared to the duration of a computation offloading session. For instance, considering users in a stadium, a conference center or a lecture hall, their positions can be assumed to be static for the duration of a computation offloading session. Moreover, we assume here that the network is connected, i.e., for each node, there exists at least one route to the AP. This is because if the network were not connected during a given computation offloading session, the nodes without connection to the server could not take part in the computation offloading session anyhow and could therefore be neglected when taking offloading decisions. Finally, we assume that the computation offloading session is associated with an a priori given routing table, i.e., we assume that some existing routing protocol for wireless ad-hoc networks [RT99] is used to fix a unique route from each node in the network to the AP, depending on the current positions of the nodes and the current channel conditions. We make this assumption since we want our model and approach to work on top of any routing protocol, which has as output a routing

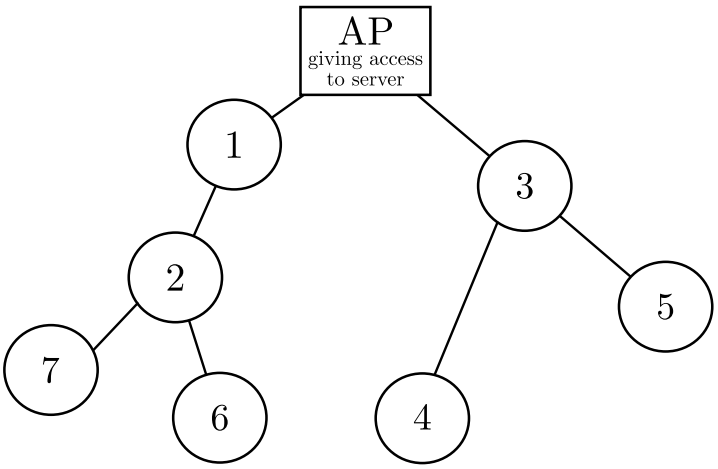


Figure 3.2. An exemplary multi-hop network in graph representation.

table with unique routes from the nodes in the multi-hop network to the AP. According to the routes stored in the routing table, nodes close to the server may have a direct one-hop connection to the AP, while other nodes may have to access the AP via a unique multi-hop route.

Based on the given routing table, the current state of a multi-hop network may be represented as a graph, by setting the AP and nodes 1 to N as vertices of the graph and by including an edge between two vertices if and only if a link exists between the corresponding nodes in the routing table. Since each node has a unique route to the AP, the resulting graph is a rooted tree, where the root represents the AP. We refer to the internal nodes of this tree as *relay nodes* and denote the set of all relay nodes by $\mathcal{R}$. Using basic tree properties [Fou10], it follows that the number $R := |\mathcal{R}|$ of relay nodes satisfies $R \leq N - 1$. We refer to the leaf nodes of the tree as *non-relay nodes*. Moreover, for a node $n \in \{1, ..., N\}$, we denote the set of relay nodes on the multi-hop route from node n to the AP by $\mathcal{R}_n$. The nodes on route $\mathcal{R}_n$ from node n to the AP are called node n's *predecessors*. The direct predecessor of node n is called node n's *parent*. Those nodes of which node n is a predecessor, are called node n's *successors*. Figure 3.2 shows an exemplary multi-hop network in graph representation with an AP giving server access and with 7 nodes. Nodes 1, 2 and 3 are relay nodes, i.e., $\mathcal{R} = \{1, 2, 3\}$. Nodes 4, 5, 6 and 7 are non-relay nodes. The route from node 6 to the AP is $\mathcal{R}_6 = \{2, 1\}$, i.e., nodes 2 and 1 are predecessors of node 6. Node 2 is the parent of nodes 6 and 7. Since node 3 does not have any predecessor node, the route from node 3 to the AP is $\mathcal{R}_3 = \emptyset$. The successors of node 3 are nodes 4 and 5.

3.3.3 Context Model

We define a context model to characterize the devices' computation tasks. In a computation offloading session, each node n, $n = 1, ..., N$, is assumed to have a non-splittable task suitable for computation offloading [KL10, KLLB13]. A task is described based on two properties, which characterize its data requirements. On the one hand, computing the task of node n requires processing a specific number L_n of CPU cycles [KL10, KLLB13]. On the other hand, the task is characterized by the number B_n of bits that need to be transmitted when offloading the task for remote processing [KL10, KLLB13]. The ratio $\frac{B_n}{L_n}$ between the number B_n of bits needed for transmission and the number L_n of CPU cycles needed for computation is application-specific [MN10]. We refer to the number B_n of bits needed for transmission and the number L_n of CPU cycles as *task context*. As will be discussed in the sequel, the energy consumption for local computing as well as computation offloading depends on the task context. Therefore, the task context needs to be taken into account for offloading decisions.

3.3.4 Model of Energy Consumption for Task Processing and Transmission

We consider network energy minimization as performance criterion. The goal is hence to select the best subset of nodes for computation offloading, such that the overall energy spent in the system is minimized. Hence, an energy consumption model for communication and computation is needed, which can be used to compute for each task the required energy for computing the task locally and the required energy for transmitting the task to the server. Next, we formulate such a model based on existing tractable models for energy consumption in mobile devices and for wireless communication. Note that we aim at minimizing the energy consumption of the mobile devices, and hence do not consider the energy consumed by the resource-rich server for task processing.

On the one hand, depending on the processor speed M_n (in cycles / s) and the processing power $P_{\mathrm{C},n}$ (in W) of node n, the *energy per CPU cycle* $e_{\mathrm{C},n}$ (in J / cycle) for local computing at node n is according to [KL10] given by

$$e_{\mathrm{C},n} = \frac{P_{\mathrm{C},n}}{M_n}. \tag{3.1}$$

Therefore, depending on the number L_n of CPU cycles, if node n computes its task locally, the amount $E_{\mathrm{C},n}$ of energy consumed by node n is according to [KL10] given by

$$E_{\mathrm{C},n} = e_{\mathrm{C},n} L_n. \tag{3.2}$$

On the other hand, depending on the transmit power $P_{\mathrm{T},n}$ (in W) of node n, the bandwidth b_n of node n, the complex channel coefficient h_n from node n to its parent, and the noise power σ_n^2 at node n, the *energy per bit* $e_{\mathrm{T},n}$ (in J / bit) node n consumes for data transmission can be approximated using Shannon's formula [TV05] as

$$e_{\mathrm{T},n} = \frac{P_{\mathrm{T},n}}{b_n \log_2\left(1 + \frac{P_{\mathrm{T},n} \cdot |h_n|^2}{\sigma_n^2}\right)}. \tag{3.3}$$

Hence, depending on the number B_n of bits, if node n uses computation offloading, the amount $E_{\mathrm{T},n}^n$ of energy node n consumes is according to [KL10] given by

$$E_{\mathrm{T},n}^n = e_{\mathrm{T},n} B_n \tag{3.4}$$

when transmitting its own task to its parent. In addition, node n's predecessors on route $\mathcal{R}_n$ to the AP consume energy when relaying node n's task. Depending on the number B_n of bits, the energy $E_{\mathrm{T},n}^r$ consumed by a predecessor node $r \in \mathcal{R}_n$ is given by

$$E_{\mathrm{T},n}^r = e_{\mathrm{T},r} B_n, \tag{3.5}$$

where $e_{\mathrm{T},r}$ is the corresponding energy per bit consumed by node r for data transmission, given analogously as in (3.3), by replacing in (3.3) each n with an r. In sum, if node n uses computation offloading, the total amount $E_{\mathrm{T},n}$ of energy spent in the network is hence given by

$$E_{\mathrm{T},n} = E_{\mathrm{T},n}^n + \sum_{r \in \mathcal{R}_n} E_{\mathrm{T},n}^r. \tag{3.6}$$

As in [KL10, HWN12, Che15], we neglect the energy consumed on the feedback link from server to node, as the amount of feedback information for transmission of the result is small in many applications. However, our model may be extended to include the energy consumption of the feedback link as well. In this case, the structure of the problem would remain the same.

Since the nodes in the network dispose of limited energy resources, for a computation offloading session, the amount $E_{\mathrm{prov},n}$ (in J) of energy which node n provides for this

session needs to be taken into account. While the values $E_{\mathrm{C},n}$ and $E_{\mathrm{T},n}$ depend on the network topology and the characteristics of the task, the amount $E_{\mathrm{prov},n}$ of energy provided by node n may in general be chosen arbitrarily. In the sequel, we assume that each node n provides at least enough energy such that both options for its own task are feasible, i.e., the provided energy of node n is sufficient for computing task n locally or for submitting task n to node n's predecessor in the route $\mathcal{R}_n$ to the AP. Formally, this holds if

$$E_{\mathrm{prov},n} \geq \max(E_{\mathrm{C},n}, E^n_{\mathrm{T},n}) \text{ for all } n = 1, ..., N. \tag{3.7}$$

By assuming that each node is capable of computing its task locally and transmitting its task to its parent, we discard trivial cases, in which a node could not choose from the two options offloading and local computing due to its own battery constraints.

3.3.5 Architecture of Decision Making

We propose a *centralized architecture of decision making*, cf. Section 2.2.3, where the offloading decisions are taken by a central entity, for example, by a controller in the AP. We use a centralized architecture since the goal of minimizing the energy spent in the overall network requires to jointly optimize the non-trivially coupled offloading decisions of the nodes in the network. Clearly, when designing the corresponding algorithm for centralized decision making, it needs to be ensured that the amount of information to be collected centrally is kept low.

3.3.6 Action Model

The central entity should decide for each node n, $n = 1, ..., N$, whether the node should compute its task locally or, if enough resources are available at relay nodes, whether node n should use computation offloading by transmitting the task to the server for remote processing. In this case, the corresponding number B_n of bits needs to be transmitted to the AP via multi-hop route $\mathcal{R}_n$. The action with respect to node n is formalized by the binary variable y_n, where

$$y_n := \begin{cases} 1, & \text{if node } n \text{ transmits its task to the server} \\ 0, & \text{if node } n \text{ computes its task locally.} \end{cases} \tag{3.8}$$

The vector of actions with respect to all nodes in the network is given by $\mathbf{y} := [y_1, ..., y_N]$. We denote the set of nodes transmitting their tasks to the server

by $\mathcal{N}_{\mathrm{T}} := \{n \in \{1, ..., N\} : y_n = 1\}$ and the set of nodes computing their tasks locally by $\mathcal{N}_{\mathrm{C}} := \{n \in \{1, ..., N\} : y_n = 0\}$. In a computation offloading session, after the central entity has selected the action of each node and informed the nodes about its decisions, the nodes from set $\mathcal{N}_{\mathrm{T}}$ transmit their tasks to the server, where we assume that a scheduling scheme is used, which allows the nodes to transmit their tasks in an interference-free fashion. Such interference-free transmissions may be achieved by giving nodes orthogonal resources, such as individual sub-carriers using *frequency-division multiple access* (FDMA) or fractions of time using *time-division multiple access* (TDMA) [Gol05]. The server then processes the received tasks in parallel and sends the results back to the corresponding nodes.

3.4 Problem Formulation

In this section, using the models from Sections 3.3.2 - 3.3.6, we take an *optimization-based approach* by formulating the problem of context-aware computation offloading for energy minimization in multi-hop wireless networks as a network-wide optimization problem to be solved by a central entity, for example, by a controller in the AP. We use an optimization-based approach, cf. Section 2.3.2, since based on the existing tractable task and energy models for mobile devices [MN10], as given in Section 3.3.4, the parameters appearing in the problem may be assumed to be known.

Aiming at the global goal of minimizing the total energy spent in the network, a network cost function needs to be formulated, which computes the total energy spent in the network for local computing and transmission to the server as a function of the nodes' actions. Based on the energies $E_{\mathrm{C},n}$ and $E_{\mathrm{T},n}$ computed in (3.2) and (3.6), the total energy $E_{\mathrm{net}}(\mathbf{y})$ spent in the network for local computing and transmission to the server as a function of the actions $\mathbf{y}$ of all nodes is given by

$$\begin{aligned} E_{\mathrm{net}}(\mathbf{y}) &= \sum_{n=1}^{N} \left(y_n E_{\mathrm{T},n} + (1 - y_n) E_{\mathrm{C},n}\right) \\ &= \sum_{n=1}^{N} y_n (E_{\mathrm{T},n} - E_{\mathrm{C},n}) + \sum_{n=1}^{N} E_{\mathrm{C},n}. \end{aligned} \tag{3.9}$$

The network cost function in (3.9) is a linear function in the action vector $\mathbf{y}$. Moreover, since the second term $\sum_{n=1}^{N} E_{\mathrm{C},n}$ is independent of $\mathbf{y}$, it can be neglected when solving an optimization problem with objective function $E_{\mathrm{net}}(\mathbf{y})$.

The nodes' limited energy resources may impose constraints on possible actions. On the one hand, the limited energy resources of a non-relay node $n \notin \mathcal{R}$ only need to be

sufficient for locally processing its own task or transmitting its own task to its parent, which by the assumption in (3.7) is always the case. On the other hand, assuming that a relay node $r \in \mathcal{R}$ provides a total amount $E_{\text{prov},r}$ of energy for a computation offloading session and assuming that the node should at least ensure that its own standard action of local computing is always possible, regardless of the actions of other nodes in the network, node r should reserve an amount of $E_{\text{C},r}$ for itself. Therefore, if node r reserves an amount of $E_{\text{C},r}$ for itself, it may spend an additional amount $E_{\text{prov},r} - E_{\text{C},r} \geq 0$ of energy for relaying tasks of successor nodes. Hence, the energy $E_{\text{prov},r} - E_{\text{C},r} \geq 0$ provided by any relay node $r \in \mathcal{R}$ for a computation offloading session restricts the offloading decisions at successor nodes. This can be formalized as follows:

$$\sum_{\{n:r\in\mathcal{R}_n\}} y_n E^r_{\text{T},n} \leq E_{\text{prov},r} - E_{\text{C},r}. \tag{3.10}$$

The constraint in (3.10) is linear in the action vector $\mathbf{y}$.

Employing the network cost function in (3.9) as objective function and taking into account the energy constraints at relay nodes in (3.10), the optimal actions minimizing the network cost may be obtained by solving the following ILP problem with binary variables:

$$\begin{aligned} \min \ & \sum_{n=1}^{N} y_n (E_{\text{T},n} - E_{\text{C},n}) \\ \text{s.t.} \ & \sum_{\{n:r\in\mathcal{R}_n\}} y_n E^r_{\text{T},n} \leq E_{\text{prov},r} - E_{\text{C},r} \text{ for } r \in \mathcal{R} \\ & y_n \in \{0,1\} \text{ for } n = 1, ..., N. \end{aligned} \tag{3.11}$$

Here, the number of constraints corresponds to the number $R = |\mathcal{R}|$ of relay nodes and is hence topology-dependent. Moreover, based on (3.2), (3.4) and (3.5), Problem (3.11) depends on the task context.

3.5 Problem Analysis

3.5.1 Equivalence to Multi-Dimensional Knapsack Problem

In this section, Problem (3.11) is analyzed. As we show below, Problem (3.11) corresponds to a specific type of ILP problem with binary variables, namely, to the multi-dimensional knapsack problem, which was introduced in Section 2.3.2.4.

Proposition 3.1. *Problem* (3.11) *corresponds to a multi-dimensional knapsack problem.*

The proof of Proposition 3.1 can be found in Appendix A.1. The proof works straightforward by rewriting Problem (3.11) into the format of a multi-dimensional knapsack problem as given in (2.3) in Section 2.3.2.4.

From Proposition 3.1 it follows by the discussion in Section 2.3.2.4 that Problem (3.11) belongs to the complexity class of NP-hard optimization problems, such that it is widely believed that there exists no polynomial-time algorithm for its solution. Moreover, by Section 2.3.2.4, it is even believed that no fully polynomial time approximation scheme exists for Problem (3.11). Therefore, we propose a heuristic algorithm for Problem (3.11) in Section 3.6.

3.5.2 Feasibility

Next, we investigate the feasibility of Problem (3.11). Problem (3.11) is always feasible since it is always possible that all nodes compute their tasks locally. This can be easily checked by inserting the corresponding action $\mathbf{y} = [0, ..., 0]$ into Problem (3.11), which yields a feasible solution.

3.5.3 Variable Reduction

Here, we discuss a possible pre-processing step which can be used in certain cases to reduce the problem size. This is due to the fact that under certain conditions, the optimal action of a node may immediately be found, without solving the overall Problem (3.11). In detail, this is the case when the parameters corresponding to the node in Problem (3.11) do not satisfy the nonnegativity assumptions, which hold without loss of generality for multi-dimensional knapsack problems as stated in Section 2.3.2.4. For such cases, one may apply rules of how to fix decision variables corresponding to negative parameters to their optimal values in multi-dimensional knapsack problems [KPP04]. Applying these rules in a pre-processing step reduces the number of variables appearing in Problem (3.11) and hence the problem size. In the sequel, we give two rules of variable reduction in Problem (3.11), which we derived from the rules of variable reduction in multi-dimensional knapsack problems [KPP04]. The following rules may be used as pre-processing steps before solving Problem (3.11).

- Rule 1: If for any node n, $E_{\mathrm{T},n} \geq E_{\mathrm{C},n}$ holds, i.e., less energy is consumed when computing node n's task locally compared to the overall energy consumed when transmitting node n's task to the server via the multi-hop route, then node n should compute locally. This is because transmitting to the server would not decrease the consumed energy in the network and possibly consumes energy at relay nodes, which could otherwise be used for other task transmissions. Thus, the optimal action for node n is $y_n^* = 0$.
- Rule 2: If for any relay node $r \in \mathcal{R}$, $E_{\mathrm{prov},r} = E_{\mathrm{C},r}$ holds, i.e., node r reserves its provided energy resources for computing locally, then each successor node of relay node r has to compute locally. Thus, for each node n with $r \in \mathcal{R}_n$ the only possible (and therefore optimal) action is $y_n^* = 0$.

3.5.4 Decomposition

Depending on the network topology, Problem (3.11) may be decomposed into several smaller problems. For this purpose, consider a multi-hop network in its graph representation. As described in Section 3.3.2, the resulting graph is a rooted tree, whose root represents the AP giving access to the server. In order to decompose Problem (3.11), this tree is partitioned into subtrees, where each subtree consists of one child node of the AP and all its successor nodes. Then, Problem (3.11) can be decomposed into one sub-problem per subtree since only decisions of nodes on the same subtree are coupled by energy constraints of common relay nodes. As an example, Figure 3.3 shows how the exemplary multi-hop network from Figure 3.2 is partitioned into two subtrees, as indicated by the dashed and the dotted line. In this example, the corresponding Problem (3.11) can be decomposed into two smaller problems, one for each of the two subtrees.

3.5.5 Analytical Results for Special Topologies

In the following, we prove analytical results for Problem (3.11) in case of special topologies. We start with a star topology, cf. Figure 3.4, which establishes the connection of Problem (3.11) to computation offloading in single-hop networks. Next, by considering a line topology, cf. Figure 3.5, the impact of a non-decomposable topology with maximum number of relay nodes on the benefit of computation offloading is investigated. In this way, we derive conditions with respect to the topology, under which computation offloading is beneficial in multi-hop networks.

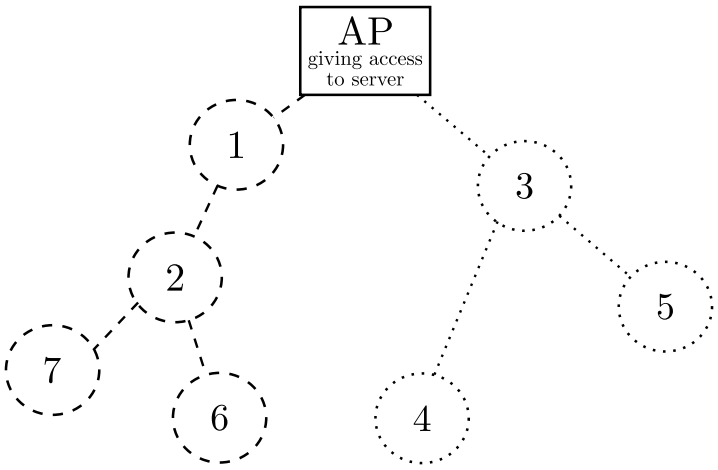


Figure 3.3. Partition of an exemplary multi-hop network in graph representation into subtrees.

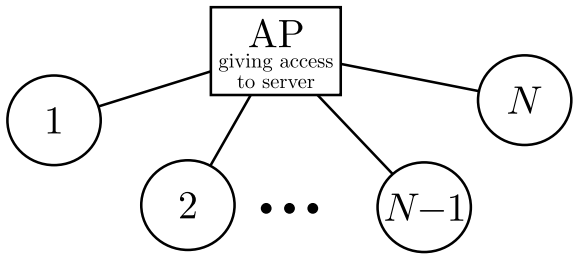


Figure 3.4. An exemplary network in star topology.

Consider a network of N nodes in *star* topology, i.e., a single-hop network with a root and N leaves. Figure 3.4 shows an example of a network in star topology. Since there is no relay node in a star topology, the corresponding energy minimization Problem (3.11) does not contain any energy constraints. Thus, in this case, Problem (3.11) may be optimally solved node-wise (i.e., decomposed into one sub-problem per node, cf. Section 3.5.4) by determining for each node n individually which is the less energy consuming action based on the values of $E_{T,n}$ and $E_{C,n}$. Hence, the following statement holds.

Proposition 3.2. *Consider Problem* (3.11) *in the case of a star topology. The globally optimal actions* y_n^*, $n = 1, ..., N$, *are given by*

$$y_n^* = \begin{cases} 1, & \text{if } E_{\mathrm{T},n} < E_{\mathrm{C},n} \\ 0, & \text{if } E_{\mathrm{T},n} \geq E_{\mathrm{C},n}. \end{cases} \tag{3.12}$$

Next, consider a network of N nodes in *line* topology, i.e., a network consisting of one single rooted branch whose nodes can be labeled according to their hop distance to

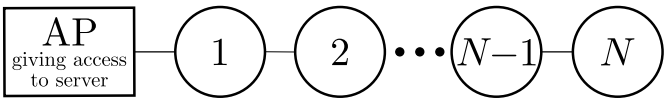


Figure 3.5. An exemplary network in line topology.

the AP from 1 to N. An example of a network in line topology is given in Figure 3.5. In this case, the number R of relay nodes satisfies $R = N - 1$. Therefore, the corresponding Problem (3.11) contains the highest possible number of energy constraints. However, since the tree consists of one single branch, the energy constraints have a special structure, which allows to give an analytical result of Problem (3.11) in case of a homogeneous network, where the energies per node are equal.

Proposition 3.3. *Consider Problem* (3.11) *in the case of a line topology in a homogeneous network, where there exist constants* $E_{\mathrm{C}} \geq 0, E_{\mathrm{link}} \geq 0, E_{\mathrm{prov}} \geq 0$ *with*

$$E_{\mathrm{C},n} = E_{\mathrm{C}} \qquad \textit{for all} \;\; n = 1, ..., N \tag{3.13}$$

$$E^r_{\mathrm{T},n} = E_{\mathrm{link}} \qquad \textit{for all} \;\; n, r = 1, ..., N \;\; \textit{with} \;\; n \geq r \tag{3.14}$$

$$E_{\mathrm{prov},n} = E_{\mathrm{prov}} \qquad \textit{for all} \;\; n = 1, ..., N-1. \tag{3.15}$$

Then, the optimal actions y^*_n, $n = 1, ..., N$, *are given by*

$$y^*_n = \begin{cases} 1, & \textit{if } n < \frac{E_{\mathrm{C}}}{E_{\mathrm{link}}} \;\textit{ and }\; n \leq \frac{E_{\mathrm{prov}} - E_{\mathrm{C}}}{E_{\mathrm{link}}} + 1 \\ 0, & \textit{else.} \end{cases} \tag{3.16}$$

The proof of Proposition 3.3 can be found in Appendix A.2. The idea of the proof is as follows. Two cases are distinguished and for each case, it is shown that the optimal actions are given by (3.16). First, the case of $E_{\mathrm{prov}} = E_{\mathrm{C}}$ is considered. This is a special case in which it is easy to show that the optimal actions are given by (3.16). Then, the case of $E_{\mathrm{prov}} > E_{\mathrm{C}}$ is considered. Due to the special structure of the line topology, it is possible to simplify the formula for the total amount $E_{\mathrm{T},n}$ of energy spent in the network when node n uses computation offloading, as given in (3.6). Together with the pre-processing Rule 1 from Section 3.5.3, this yields that $n < \frac{E_{\mathrm{C}}}{E_{\mathrm{link}}}$ has to be satisfied for any node n whose optimal action it is to use computation offloading. Moreover, due to the special structure of the line topology, where the set of successor nodes of a relay node is given by all nodes which have higher hop distance to the AP, the constraints in Problem (3.11) can be simplified. Using this fact in combination with a concept for knapsack problems called dominance [KPP04], one can prove that computation offloading is the optimal action for any node n which has $n \leq \frac{E_{\mathrm{prov}} - E_{\mathrm{C}}}{E_{\mathrm{link}}} + 1$. Combining these results, it follows that the optimal actions are given by (3.16).

Proposition 3.3 illustrates that in a homogeneous line topology, computation offloading is optimal for any node n, whose hop distance n to the AP satisfies the two conditions $n < \frac{E_{\mathrm{C}}}{E_{\mathrm{link}}}$ and $n \leq \frac{E_{\mathrm{prov}} - E_{\mathrm{C}}}{E_{\mathrm{link}}} + 1$. Hence, combining the two conditions yields a threshold. All nodes whose hop distance to the AP is smaller than the threshold will use computation offloading. All nodes whose hop distance to the AP is larger than the threshold will compute locally.

Note that for topologies consisting of several parallel homogeneous lines to the AP, Problem (3.11) is decomposable to each of these lines based on the result from Section 3.5.4. Hence, in this case, Proposition 3.3 may be applied to each of these individual lines.

3.6 Proposed Algorithm

In this section, using an optimization-based approach and a centralized architecture of decision making, we propose a *context-aware greedy heuristic algorithm for computation offloading in multi-hop networks* to tackle the energy minimization Problem (3.11). Using the proposed algorithm, a central entity, e.g., a controller in the AP, may take offloading decisions based on centrally collected information about network conditions and task context.

Since Problem (3.11) corresponds to a multi-dimensional knapsack problem by Proposition 3.1, we propose to use the primal greedy heuristic for multi-dimensional knapsack problems [KPP04] for tackling Problem (3.11). The primal greedy heuristic is a centralized polynomial-time algorithm. In the context of computation offloading in multi-hop networks, the main idea of this algorithm is as follows.

After pre-processing based on Rules 1 and 2 from Section 3.5.3, the nodes are sorted in decreasing order of a carefully chosen efficiency measure, which will be detailed below. Then, starting from the node with the highest efficiency, the algorithm adds one node at a time to the set of nodes which offload their tasks, but only, if this does not violate any of the energy constraints at relay nodes. If an energy constraint is violated, the node is added to the set of nodes which locally compute their tasks. The critical point of the algorithm is the choice of efficiency measure. Here, we select an efficiency measure [Dob82, KPP04] that takes into account (i) how much energy is saved when node n is chosen for computation offloading compared to when node n computes its

task locally and (ii) which portions of energy provided by relay nodes it requires:

$$\text{eff}_n := \frac{E_{\text{C},n} - E_{\text{T},n}}{\sum_{r \in \mathcal{R}_n} \frac{E^r_{\text{T},n}}{E_{\text{prov},r} - E_{\text{C},r}}}. \tag{3.17}$$

The pseudocode of the context-aware greedy heuristic algorithm for Problem (3.11) is given in Algorithm 3.1. Since Problem (3.11) is decomposable to subtrees, as described in Section 3.5.4, Algorithm 3.1 may be applied separately to each of the individual subtrees.

Algorithm 3.1 Context-Aware Greedy Heuristic Algorithm

Input: Problem (3.11)
Pre-process Problem (3.11) according to Rules 1 and 2 from Section 3.5.3
Let $y_1, ..., y_{\tilde{N}}$ be the variables not fixed in pre-processing
for $n = 1, ..., \tilde{N}$ **do**
    Initialize $y_n := 0$
    $\text{eff}_n := \frac{E_{\text{C},n} - E_{\text{T},n}}{\sum_{r \in \mathcal{R}_n} \frac{E^r_{\text{T},n}}{E_{\text{prov},r} - E_{\text{C},r}}}$
end for
Sort efficiencies eff_n into decreasing order and save
    into vector ord := indices of ordered efficiencies
for $j = 1, ..., \tilde{N}$ **do**
    $y_{\text{ord}(j)} := 1$
    if decision vector **y** not feasible **then**
        $y_{\text{ord}(j)} := 0$
    end if
end for

3.7 Properties of Proposed Algorithm

3.7.1 Performance Guarantees for Special Topologies

Here, we give performance guarantees of the proposed algorithm in case of certain special topologies. Specifically, we show that the proposed algorithm automatically selects the globally optimal actions for networks in star and line topology.

We start with networks in star topology, as introduced in Section 3.5.5.

Proposition 3.4. *Consider Problem* (3.11) *in the case of a star topology. Then, the context-aware greedy heuristic algorithm in Algorithm 3.1 always selects the globally optimal actions according to* (3.12).

The proof of Proposition 3.4 can be found in Appendix A.3. The proof is straightforward by checking the steps of Algorithm 3.1 in case of a star topology, which yields that exactly the globally optimal actions according to (3.12) are selected.

Next, we consider networks with homogeneous line topologies, as introduced in Section 3.5.5.

Proposition 3.5. *Consider Problem* (3.11) *in the case of a line topology in a homogeneous network as introduced in Proposition 3.3. Then, the context-aware greedy heuristic algorithm in Algorithm 3.1 always selects the globally optimal actions according to* (3.16).

The proof of Proposition 3.5 can be found in Appendix A.4. The idea of the proof is as follows. Like in the proof of Proposition 3.3, two cases are distinguished. First, the case of $E_{\text{prov}} = E_{\text{C}}$ is considered, which is a special case in which it is easy to show that Algorithm 3.1 selects exactly the optimal actions in (3.16) from Proposition 3.3. Then, the case of $E_{\text{prov}} > E_{\text{C}}$ is considered. Due to the special structure of the line topology, it is possible to simplify the formula for the total amount $E_{\text{T},n}$ of energy spent in the network when node n uses computation offloading, as given in (3.6). Together with the fact that Algorithm 3.1 uses Rule 1 from Section 3.5.3 during pre-processing, this yields that $n < \frac{E_{\text{C}}}{E_{\text{link}}}$ has to be satisfied for any node n which Algorithm 3.1 considers as candidates for computation offloading. Moreover, due to the special structure of the line topology, one can show that Algorithm 3.1 sorts the nodes according to their hop distance to the AP. Then, Algorithm 3.1 selects nodes for computation offloading according to their hop distance, starting with node 1, as long as this does not violate any of the energy constraints. Specifically, one can show that Algorithm 3.1 selects nodes with hop distance $n \leq \frac{E_{\text{prov}} - E_{\text{C}}}{E_{\text{link}}} + 1$ for computation offloading. This shows that Algorithm 3.1 selects exactly the globally optimal actions according to (3.16).

3.7.2 Computational Complexity

Here, we analyze the computational complexity of the proposed algorithm as a function of the number N of nodes and the number R of relay nodes in the network. For this purpose, we identify the most computationally expensive procedures in the algorithm. Note that the computational complexity is described based on the O-notation, which can be used to characterize the limiting behavior of a function by giving an asymptotic upper bound on its growth rate [CLRS09]. In Line 2 of Algorithm 3.1, the pre-processing according to Rules 1 and 2 has a computational complexity that

grows as $O(R+N)$ since at most every node $n \in \mathcal{N}$ and every relay node $r \in \mathcal{R}$ need to be considered once. Computing the efficiency measure of N nodes in lines 4-7 has a computational complexity that grows as $O(NR)$. This is because for each node n, each of its relay nodes on the route to the AP needs to be considered once, and there are at most R relay nodes on this route. Sorting at most N values in line 8 has a computational complexity that grows as $O(N \log N)$ [CLRS09]. Finally, in lines 9-14, checking for at most each node $n \in \mathcal{N}$, and for at most each relay node $r \in \mathcal{R}$, whether the remaining battery of the relay node r is sufficient to add node n to the set of nodes using computation offloading has a complexity that grows as $O(NR)$. This is because each of the N nodes has to be considered once in combination with each of the R relay nodes. Overall, the computational complexity of Algorithm 3.1 is hence $O(NR + N \log N))$. Therefore, the proposed algorithm is a polynomial-time algorithm with a computational complexity that grows, depending on the ratio between relay and non-relay nodes, at most quadratically as a function of the number N of nodes in the network.

3.7.3 Communication Requirements

The context-aware greedy heuristic algorithm is based on a centralized architecture of decision making, where a central entity takes offloading decisions based on centrally collected information. Here, we investigate the communication overhead of the proposed algorithm due to its centralized approach. In detail, we compute the communication requirements for collecting the information required for decision making under the assumptions that the central entity is a controller located in the AP and that the central controller knows the routing table, which contains the multi-hop route $\mathcal{R}_n$ from each node n to the AP. In order to take the offloading decisions based on the context-aware greedy heuristic algorithm, the controller needs to compute the efficiency measure eff_n in (3.17) for each node $n = 1, ..., N$. Therefore, for a node n, the controller needs to know the values $E_{C,n}$, $E_{T,n}$, $E^r_{\text{T},n}$, $E_{\text{prov},r}$ and $E_{\text{C},r}$ for each of node n's relay node $r \in \mathcal{R}_n$.

The following procedure is proposed in order to minimize the required communication exchange between the nodes and the controller: Each node n (i) determines its parameters $e_{C,n}$ and $e_{T,n}$ according to (3.1) and (3.3), (ii) determines its provided energy $E_{\text{prov},n}$, (iii) retrieves task context information B_n and L_n and (iv) computes $E_{C,n}$ based on (3.2). Then, node n sends the values $e_{T,n}$, B_n, $E_{C,n}$, $E_{\text{prov},n}$ to the controller, by transmitting them via the multi-hop route $\mathcal{R}_n$ to the AP. Based on the received values, the controller may then compute the missing values $E^r_{\text{T},n}$ for any $r \in \mathcal{R}_n$, $n = 1, ..., N$, and $E_{T,n}$, $n = 1, ..., N$, using (3.4), (3.5) and (3.6). Subsequently, the controller applies the context-aware greedy heuristic algorithm and thereby takes the offloading decisions.

In the above procedure, each node n needs to transmit four scalar variables to the AP via its multi-hop route $\mathcal{R}_n$. Compared to the typical sizes of candidate tasks for computation offloading (e.g., 420 kbyte for a face recognition application in [SMF+12]), the signalling overhead due to centralized decision making is hence small.

3.8 Numerical Results

3.8.1 Simulation Setup

We evaluate the proposed algorithm and we analyze the benefit of computation offloading in multi-hop networks with respect to different topologies, system parameters and task contexts based on simulations. We simulate networks of $N = 20$ nodes with star, line as well as random topologies. The results for random topologies are obtained by simulating 100 random trees and averaging the results. We fix the following parameters homogeneously for each node n. For task sizes, the number of CPU cycles is set to $L_n = 1000\,\text{Mcycles}$ [Che15] and the number B_n of bits is kept variable so that we can investigate different ratios of $\frac{B_n}{L_n}$. For the energy characteristics of the devices, i.e., the energy $e_{\text{C},n}$ per bit and the energy $e_{\text{T},n}$ per CPU cycle, we take for both parameters the respective best (i.e., lowest) value which was obtained in experiments with real devices in [MN10]. This is because we want the simulations to be based on a fair ratio between the devices' energy characteristics with respect to computation and communication. In detail, for local computing, we set $e_{\text{C},n} = \frac{1}{730}\frac{\text{J}}{\text{Mcycle}}$ [MN10] and for transmission, we set $e_{\text{T},n} = \frac{1}{860}\frac{\text{J}}{\text{kbyte}}$ [MN10]. Note that based on the selected values of $e_{\text{C},n}$ $e_{\text{T},n}$, this yields an energy ratio of $\frac{e_{\text{C},n}}{e_{\text{T},n}} \approx 0.0094\frac{\text{bits}}{\text{cycle}}$ in our simulations. Concerning the energy resources spent by nodes for relaying, unless otherwise stated, we assume that each node spends an additional 100% of its own required computing energy for relaying, i.e., $E_{\text{prov},n} = 2E_{\text{C},n}$ (default value).

3.8.2 Reference Algorithms

We evaluate the proposed algorithm by comparing it with the following reference algorithms.

- The *global optimum* of Problem (3.11) gives a lower bound on the network energy consumption that the proposed algorithm can achieve. Therefore, the global

optimum may be used as benchmark. Since Problem (3.11) is an ILP problem, we use the integer programming solver Gurobi [Gur15] to find a globally optimal solution.

- In addition, we consider *pure local computing*, i.e., that all nodes in the network compute their tasks locally. Comparing the solution of pure local computing with the offloading solutions obtained by the global optimum and by the context-aware greedy heuristic algorithm allows to assess the benefit of using computation offloading in multi-hop networks.

3.8.3 Evaluation Metrics

We use the following metrics to assess the benefit of using computation offloading in multi-hop networks and to evaluate the proposed algorithm.

- We compute the total energy spent in the network and the corresponding fraction of nodes transmitting their task to the server when the offloading decisions are taken according to the algorithm. Formally, let $\mathbf{y}^A$ denote the vector of actions of all nodes selected by an algorithm A. Then, the energy spent in the network is computed as $E_{\text{net}}(\mathbf{y}^A)$ using (3.9).
- Moreover, the fraction of nodes transmitting their tasks to the server is computed as

$$\frac{|\{n \in \{1, ..., N\} : y_n^A = 1\}|}{N}, \tag{3.18}$$

where $|\cdot|$ denotes the cardinality of a set.

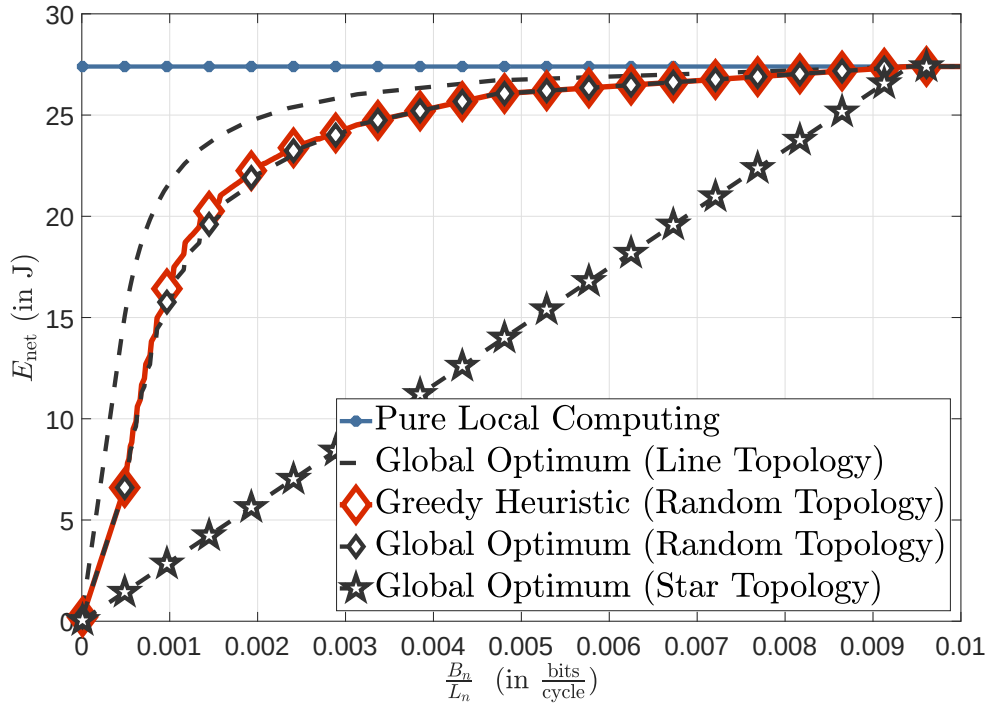
3.8.4 Results

In our simulations, we analyze the benefit of computation offloading in multi-hop networks with respect to different topologies, system parameters and task contexts and we evaluate the context-aware greedy heuristic algorithm by comparing its solutions to the global optimum. Note that each figure shown in the sequel displays results of (i) the context-aware greedy heuristic algorithm, (ii) the global optimum and (iii) pure local computing, and for each algorithm, each figure displays results under (a) random, (b) star and (c) homogeneous line topologies. Note, however, that the figures

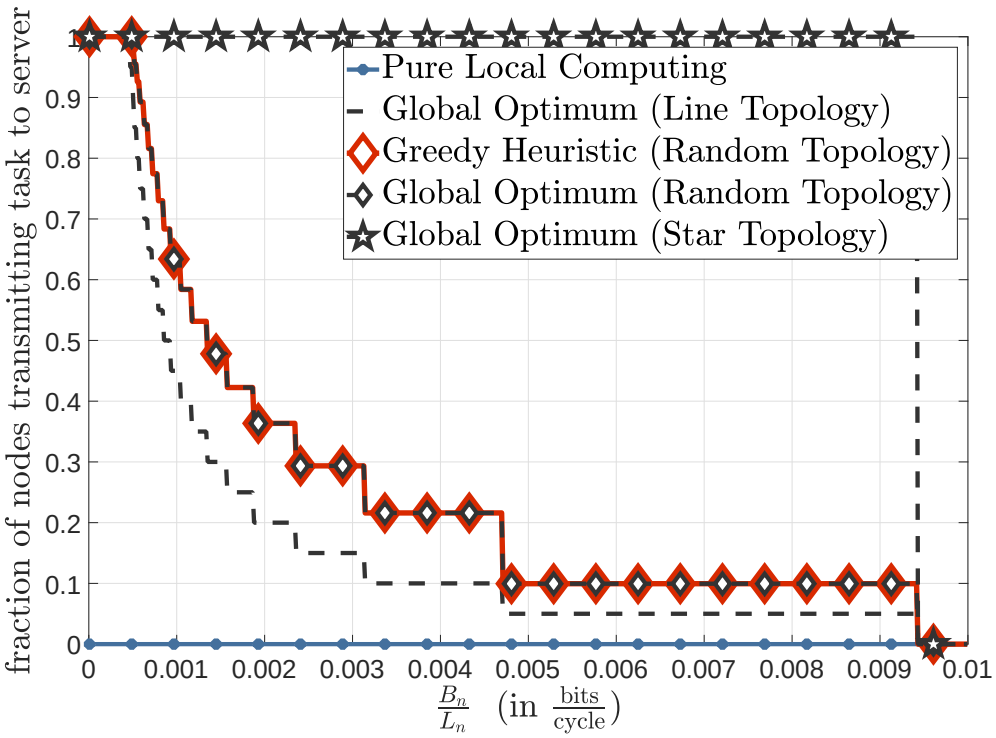
only display one curve for (i) the context-aware greedy heuristic algorithm, namely, with respect to (a) random topologies. We do not display a separate curve for (i) the context-aware greedy heuristic algorithm under the (b) star and (c) homogeneous line topologies since the context-aware greedy heuristic algorithm produces the same results as the global optimum under these topologies, such that the curves for the global optimum under these topologies actually correspond to the results of the context-aware greedy heuristic. Moreover, note that each figure only displays one curve for (iii) pure local computing. This is because pure local computing is not affected by the topology and hence produces identical results under (a) random, (b) star and (c) homogeneous line topologies.

Figure 3.6 shows the results of (i) the context-aware greedy heuristic algorithm, (ii) the global optimum and (iii) pure local computing under (a) random, (b) star and (c) homogeneous line topologies, for varying ratios $\frac{B_n}{L_n}$ between the number B_n of bits needed for transmission and the number L_n of CPU cycles needed for computation. Figure 3.6(a) gives the total energy spent in the network under varying ratios $\frac{B_n}{L_n}$. As can be seen in Figure 3.6(a), under all three types of topology and all three approaches, the energy spent in the system is a non-decreasing function of the ratio $\frac{B_n}{L_n}$. In detail, the result of pure local computing does not depend on the ratio $\frac{B_n}{L_n}$ (since L_n is fixed in our simulations) and hence stays constant for increasing $\frac{B_n}{L_n}$, and the corresponding energy spent in the system gives an upper bound for the offloading solutions obtained by the context-aware greedy heuristic algorithm and by the global optimum. When the ratio $\frac{B_n}{L_n}$ increases, i.e., transmitting a task to the server becomes more expensive in comparison to local computation, the energy spent in the system increases and converges to the same value under the offloading solutions obtained by the context-aware greedy heuristic algorithm and by the global optimum and with respect to all topologies. Convergence is reached when $\frac{B_n}{L_n} \approx 0.0094 \frac{\text{bits}}{\text{cycle}}$. Once convergence is reached, i.e., for $\frac{B_n}{L_n} \gtrapprox 0.0094 \frac{\text{bits}}{\text{cycle}}$, the energy spent in the system under the offloading solutions obtained by both the context-aware greedy heuristic algorithm and by the global optimum corresponds exactly to the energy spent in the system under pure local computing. The reason for this is that based on the assumed energy characteristics of the devices, cf. Section 3.8.1, we have an energy ratio of $\frac{e_{\text{C},n}}{e_{\text{T},n}} \approx 0.0094 \frac{\text{bits}}{\text{cycle}}$ in our simulations. Hence, when the ratio $\frac{B_n}{L_n} \gtrapprox 0.0094 \frac{\text{bits}}{\text{cycle}}$, we have $\frac{B_n}{L_n} \gtrapprox \frac{e_{\text{C},n}}{e_{\text{T},n}}$, or equivalently, $E^n_{T,n} = e_{\text{T},n} B_n \gtrapprox e_{\text{C},n} L_n = E_{C,n}$, where we used (3.2) and (3.4). This means that for any task in the network, the network's energy cost for transmitting that task to the server are as at least as high as the energy cost for local computing. Therefore, whenever $\frac{B_n}{L_n} \gtrapprox 0.0094 \frac{\text{bits}}{\text{cycle}}$, all nodes compute their tasks locally.

Next, we compare the results obtained by the different approaches under the random topology. In Figure 3.6(a), under the random topology, using the greedy offloading so-



(a) Minimum total energy E_{net} spent in network vs. ratio $\frac{B_n}{L_n}$.



(b) Fraction of nodes transmitting task to server vs. ratio $\frac{B_n}{L_n}$.

Figure 3.6. Results for $E_{\text{prov},n} = 2E_{\text{C},n}$ for homogeneous line, random and star topologies.

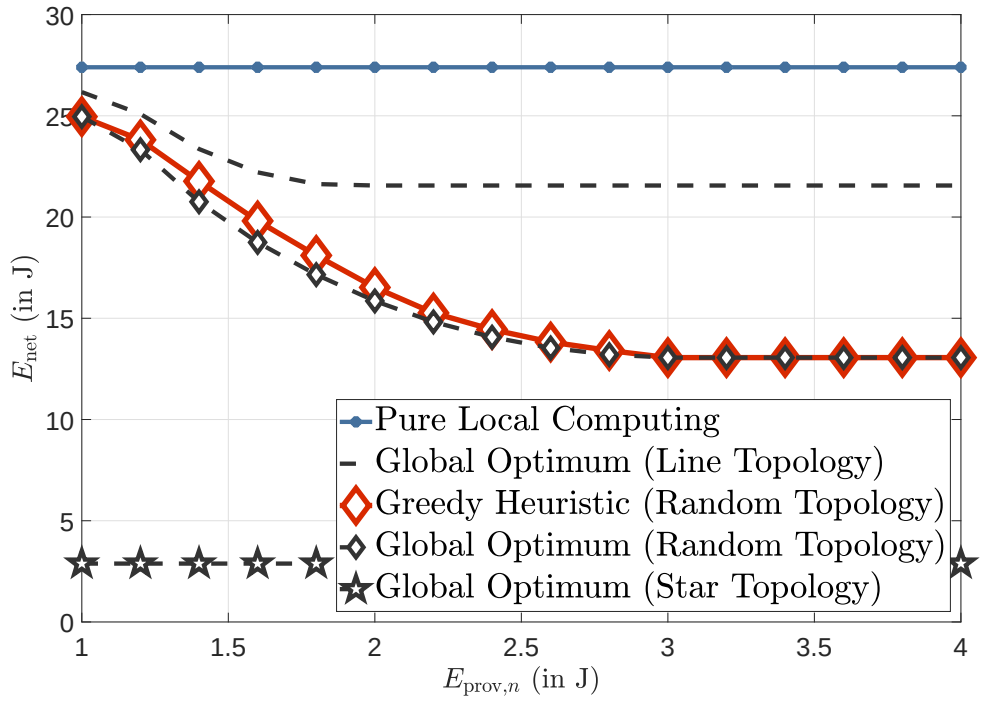
lution instead of pure local computing reduces the network energy consumption by 13% when averaging over the whole range of $\frac{B_n}{L_n}$. Moreover, under the random topology, the results obtained by the context-aware greedy heuristic algorithm lie very close to the global optimum, with a maximal deviation of less than 5% from the optimal results over the whole range of $\frac{B_n}{L_n}$. In general, as Figure 3.6(a) shows, the smaller the ratio of $\frac{B_n}{L_n}$, the more energy can be saved when using computation offloading instead of pure local computing, no matter whether the offloading solution obtained by the context-aware greedy heuristic algorithm or the global optimum is used.

Finally, we study how the underlying topology affects the benefit of computation offloading in multi-hop networks. Comparing the results of the global optimum for the different topologies in Figure 3.6(a), one can see that averaged over the whole range of $\frac{B_n}{L_n}$, the energy consumption of the star topology (i.e., a single hop network) is much smaller than that of a random topology, which itself is slightly smaller than that of the line topology. Moreover, Figure 3.6(a) shows, comparable to what has been shown in [KL10,MN10], that computation offloading noticeably pays off in single-hop networks for any ratio of $\frac{B_n}{L_n}$ as long as $\frac{B_n}{L_n} < \frac{e_{\mathrm{C},n}}{e_{\mathrm{T},n}}$ holds. This is because in single-hop networks, each node for whom transmission to the server is cheaper, uses computation offloading, as was also shown analytically in Section 3.5.5. In contrast, for multi-hop networks, Figure 3.6(a) shows that the effect of computation offloading becomes particularly apparent for very small ratios of $\frac{B_n}{L_n}$, for which $\frac{B_n}{L_n} \ll \frac{e_{\mathrm{C},n}}{e_{\mathrm{T},n}}$, or equivalently, $E^n_{T,n} \ll E_{C,n}$ holds, where we used (3.2) and (3.4). This means that computation offloading in multi-hop networks is beneficial (i) for applications with very high computation effort and very small amounts of data to be transmitted (i.e., large L_n and small B_n) and/or (ii) if the devices' energy capabilities in terms of computation are much worse than in terms of communication (i.e., large $e_{\mathrm{C},n}$ and small $e_{\mathrm{T},n}$). In addition, the benefit of computation offloading is more prominent in topologies with on average smaller hop distances to the AP. If many hops are involved, such as in a line topology, having $\frac{B_n}{L_n} \ll \frac{e_{\mathrm{C},n}}{e_{\mathrm{T},n}}$ is even more important in order to benefit from computation offloading, as can also be seen from the results in Section 3.5.5.

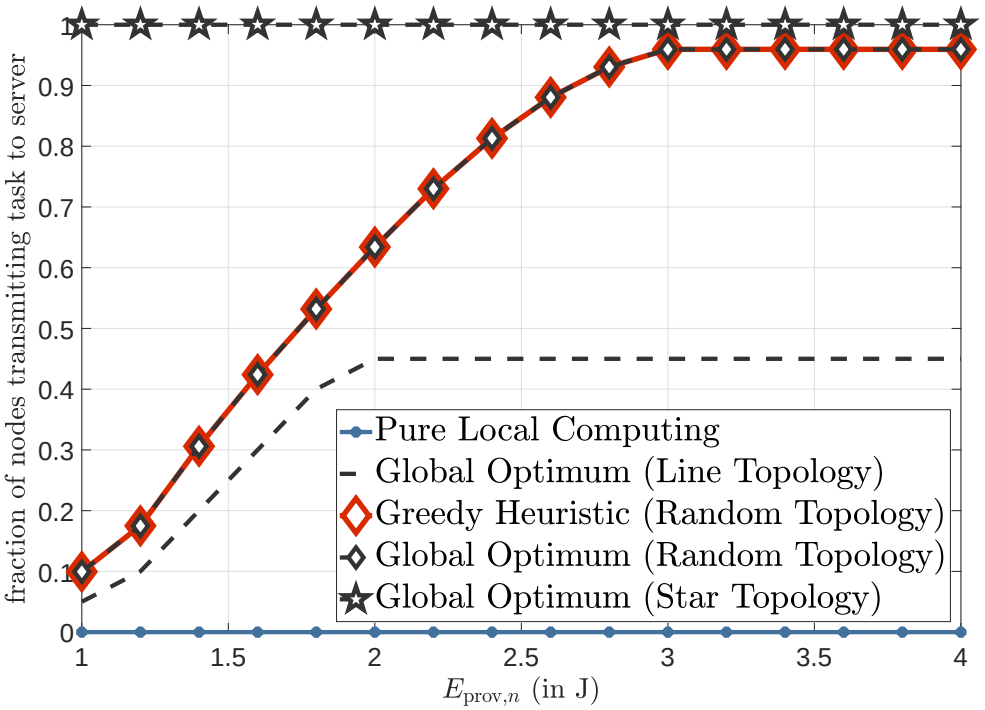
Figure 3.6(b) shows the fraction of nodes transmitting their task to the server under varying ratio $\frac{B_n}{L_n}$ between the number B_n of bits needed for transmission and the number L_n of CPU cycles needed for computation. As shown in Figure 3.6(b), under all three types of topology and all three approaches, the fraction of nodes transmitting their task to the server is a non-increasing function of the ratio $\frac{B_n}{L_n}$. For pure local computing, the fraction of nodes transmitting their task to the server is obviously zero for any value of the ratio $\frac{B_n}{L_n}$. Moreover, when the ratio $\frac{B_n}{L_n}$ increases, i.e., transmitting a task to the server becomes more expensive in comparison to local computation, the fraction of nodes transmitting their task to the server decreases under the offloading

solutions obtained by the context-aware greedy heuristic algorithm and by the global optimum and with respect to all topologies. This results from the fact that when data transmission gets more expensive, more and more nodes instead compute locally. In addition, when the ratio $\frac{B_n}{L_n}$ increases, the fraction of nodes transmitting their task to the server converges to zero under the offloading solutions obtained by the context-aware greedy heuristic algorithm and by the global optimum and with respect to all topologies. In analogy to the results discussed above for Figure 3.6(a), convergence is reached when $\frac{B_n}{L_n} \approx 0.0094 \frac{\text{bits}}{\text{cycle}}$ since then, as discussed above, for any task in the network, the network's energy cost for transmitting that task to the server are at least as high as the energy cost for local computing and therefore, all nodes compute their tasks locally. It can be observed in Figure 3.6(b), that the curves of the context-aware greedy heuristic and the global optimum under random and line topologies coincide and that they are step functions. The lines coinciding shows how close the context-aware greedy heuristic approaches the result of the global optimum. While we have seen in Figure 3.6(a), that the context-aware greedy heuristic deviates at most 5% from the global optimum in terms of the energy consumption, it actually always selects the same number of nodes to transmit their tasks to the server. The steps in the graphs reflect the impact of the energy $E_{\text{prov},n}$ which the nodes provide per computation offloading session. If there is not enough energy left for a node at one of its relay nodes to further perform computation offloading, the node is forced to compute locally, leading to an abrupt increase of the energy spent.

Next, in order to evaluate the effect of the provided energy $E_{\text{prov},n}$, we fix the task size to $B_n = 124\,\text{kbyte}$ and run simulations for varying parameter $E_{\text{prov},n}$. With $B_n = 124\,\text{kbyte}$, the energy ratio satisfies $\frac{B_n}{L_n} \approx 0.001 \frac{\text{bits}}{\text{cycle}}$, i.e., tasks are computationally expensive, but very cheap in terms of data transmission. Specifically, due to the energy ratio of $\frac{e_{\text{C},n}}{e_{\text{T},n}} \approx 0.0094 \frac{\text{bits}}{\text{cycle}}$ in our simulations, using (3.2) and (3.4), this yields $E^n_{T,n} \approx \frac{1}{10} E_{C,n}$, i.e., local computation of a task is roughly 10 times as expensive as one-hop transmission of the task. We use such a small number B_n of bits since we do not want the offloading decisions to be constrained by the task size in this particular simulation in order to see the pure effect of the provided energy $E_{\text{prov},n}$. Figure 3.7 shows the results of (i) the context-aware greedy heuristic algorithm, (ii) the global optimum and (iii) pure local computing under (a) random, (b) star and (c) homogeneous line topologies for varying $E_{\text{prov},n}$. Figure 3.7(a) gives the total energy spent in the network and Figure 3.7(b) shows the corresponding fraction of nodes transmitting their task to the server. Clearly, pure local computing and the offloading solution for star topologies is not affected by a change of $E_{\text{prov},n}$. However, Figure 3.7 shows that for random and line topologies, with increasing value of $E_{\text{prov},n}$, the energy consumption is decreasing when optimal or greedy offloading solutions are applied since more and more nodes



(a) Minimum total energy E_{net} spent in network vs. provided energy $E_{\text{prov},n}$.



(b) Fraction of nodes transmitting task to server vs. provided energy $E_{\text{prov},n}$.

Figure 3.7. Results for $B_n = 124\,\text{kbyte}$ for homogeneous line, random and star topologies.

offload their tasks to the server. The context-aware greedy heuristic algorithm again yields similar results as the global optimum, with a maximum deviation of less than 6% from the optimal result. Moreover, for both random and line topologies, the network energy consumption converges for large values of $E_{\text{prov},n}$, but to two different values. At the same time, for both random and line topologies, the fraction of nodes transmitting their tasks to the server converges for large values of $E_{\text{prov},n}$, but also to two different values. This is due to the fact that if $E_{C,n} < E_{T,n}$ holds for a node, this node will never use computation offloading, no matter how large the provided energy $E_{\text{prov},n}$ is. Hence, there exists a point after which a further increase of $E_{\text{prov},n}$ does not lead to more nodes offloading their tasks. Since the energy costs $E_{T,n}$ are especially high in a line topology due to the large number of hops, in this case, the fraction of offloading nodes is lower than in a random topology.

From our simulations, we may conclude the following. First, computation offloading in multi-hop networks is beneficial for highly computation-intensive applications with small amounts of data to be transmitted. Secondly, the effect of computation offloading strongly depends on the provided energy $E_{\text{prov},n}$. With higher amounts of provided energy, computation offloading may save more energy in the overall network, but the energy savings do not grow arbitrarily for larger values of provided energy since for some tasks, it is always cheaper to compute them locally even if more energy is provided by relay nodes. Thirdly, even though the context-aware greedy heuristic algorithm has no performance guarantee for general multi-dimensional knapsack problems, it yields very good overall results in the considered offloading scenarios, with a maximal deviation of less than 6% from the optimal results for the considered set of parameters.

3.9 Conclusions

In this chapter, we have investigated how to trade computation resources off against communication resources in wireless networks. Specifically, we have studied the problem of context-aware computation offloading for energy minimization in multi-hop wireless networks. We have proposed a general model for context-aware computation offloading, which takes into account the challenges of multi-hop networks, i.e., the need to use and share communication resources of relay nodes. Moreover, we have modeled the problem as an optimization problem and shown its equivalence to a multi-dimensional knapsack problem. We have proposed a context-aware greedy heuristic algorithm for computation offloading in multi-hop networks using a centralized architecture of decision making. Using this algorithm, a controller in the AP may take offloading decisions based on centrally collected information about network conditions and task context.

The computational complexity of the proposed algorithm has been shown to grow at most quadratically as a function of the number of nodes in the network. Moreover, the communication overhead of the proposed centralized architecture of decision making has been shown to be small. In addition, the proposed algorithm has shown very good performance in simulations under various network settings and task contexts, with a maximal deviation of less than 6% from the optimal results. On average, the offloading solution found by the proposed algorithm reduces the network energy consumption by 13% compared to the case when no computation offloading is used. Our numerical as well as analytical results have revealed that devices in multi-hop networks benefit noticeably from computation offloading for highly computation-intensive applications with small amount of data to transmit. Additionally, the outcome is strongly affected by the amount of energy provided by relay nodes, but the energy savings do not grow arbitrarily when the provided energy is increased.

Chapter 4

Caching at the Edge of Wireless Networks

4.1 Introduction

In this chapter, we study caching at the edge of wireless networks. Caching at the edge exploits *caching resources* at the edge of the wireless network in order to serve users locally with popular contents [BBD14b]. Such caching resources could be attached to *macro base stations* (MBSs) and *small base stations* (SBSs) owned by the *mobile network operator* (MNO) or they could be part of *wireless infostations* installed in public or commercial areas by either a content provider or a third party [GBMY97, IR02, BG14c, BG14a]. Caching popular content in local caches in a placement phase and locally serve the users in a delivery phase may reduce backhaul and cellular traffic and it may reduce the latency for the user [WCT+14]. In order to reduce the load on the macro cellular network as much as possible, the most popular content should be cached locally such that the number of cache hits is maximized. As described in Section 1.3.3, this requires knowledge about the popularity distribution, which is typically not available a priori [BBD14b, BBZ+15, BG14b, BG14c, BG14a, SAT+14, EBSLa14]. Moreover, local content popularity may vary according to the preferences of the mobile users connecting to a local cache over time [GALM07, ZSGK09, BSW12]. The users' preferences, in turn, may depend on their contexts [BSW12, MS10, HL05, RGZ11, Zil88, ZGC+14]. Finally, cache content placement needs to take into account the cache operator's specific objective, which may include the need for service differentiation [KLAC03, LAS04].

We hence consider the problem of maximizing the number of cache hits in a local cache at the edge of the wireless network, taking into account the following aspects:

(i) A priori, there is no knowledge available about local content popularity.

(ii) Content popularity can vary across the user population.

(iii) Content popularity can depend on the users' contexts.

(iv) The cache operator's specific objective with respect to service differentiation needs to be taken into account.

In the sequel, we propose a *machine-learning-based approach* and a *decentralized architecture of decision making*. We use a machine-learning-based approach since the content popularity is not known in advance and needs to be learned. Moreover, we use a decentralized architecture of decision making and let the controller of a local cache take local caching decisions since the content popularity at a local cache is not necessarily the same as the global content popularity and since the set of mobile users with potentially different interests in the vicinity of a local cache changes over time. In detail, we propose an online learning algorithm for context-aware proactive caching based on a *contextual multi-armed bandit* (contextual MAB) model. Using this algorithm, the controller of a local cache at the edge of the wireless network is enabled to learn context-specific content popularity online over time. This chapter presents work originally published by the author in [MAvK16, MAvK17]. Compared to [MAvK16, MAvK17], in this thesis, the regret bound is improved in its constant factors due to a new proof technique. Furthermore, in this thesis, an analysis of the computational complexity and of the communication requirements of the proposed algorithm is added. Also, in this thesis, the ideas of the mathematical proofs are additionally summarized and discussed within the main body of text, while the full mathematical proofs are given in the appendices. In addition, in this thesis, the numerical results are revised to show a better comparison of the proposed algorithm with the oracle solution, which assumes a priori knowledge about local content popularity.

The remainder of this chapter is organized as follows. Section 4.2 provides a detailed review of the state of the art on decision making for caching at the edge. In Section 4.3, we introduce the system model for context-aware proactive caching at the edge. A formal problem formulation of context-aware proactive caching for maximizing the number of cache hits under missing knowledge about content popularity is presented in Section 4.4 and it is shown that the formulated problem can be understood as a contextual MAB problem. In Section 4.5, we propose an online learning algorithm for context-aware proactive caching. In Section 4.6, properties of the proposed algorithm are discussed. In particular, an analytical upper bound on the regret of the proposed algorithm is derived, which proves that the algorithm converges to the optimal cache content placement strategy. Extensions of the proposed algorithm to practical requirements are presented in Section 4.7. The performance of the proposed algorithm is evaluated numerically in Section 4.8. Section 4.9 concludes this chapter.

4.2 State of the Art

In this section, a review of the state of the art on decision making for caching at the edge is presented. We start by shortly discussing algorithms that continuously update the cache during the delivery phase. Then, since it corresponds to the type of problem considered in this thesis, we discuss in detail the related work on cache content placement problems with a separate placement phase and a separate delivery phase.

Many practical caching systems apply caching algorithms which update the cache continuously during the delivery phase, such as the well-known *Least Recently Used* (LRU) or *Least Frequently Used* (LFU) algorithms [CI97]. While many of these algorithms are rather simple and typically do not consider future content popularity, recently, advanced algorithms updating the cache continuously during the delivery phase were proposed which also learn content popularity trends [LXvdSL16b, LXvdSL16a].

In contrast to approaches which update the cache continuously during the delivery phase, in the related work, cache content placement is also often studied for wireless caching problems with a separate placement phase and a separate delivery phase. Also the problem of cache content placement considered in this thesis is based on a placement phase and a delivery phase. Therefore, in the remainder of this section, we review and discuss related works on cache content placement for wireless caching problems with a placement phase and a delivery phase. This review complements the short review presented in Section 1.3.3 by discussing in detail the works introduced in Table 1.2.

One line of literature considers cache content placement under the assumption that *knowledge about content popularity is available in advance*. The efficiency of content delivery is increased in [GMDC13] by combining content caching at user devices and collaborative device-to-device communication. In [SGD+13], the problem of minimizing the average delay experienced by users that can be connected simultaneously to several cache-enabled SBSs is considered and an approximation algorithm for uncoded caching among SBSs is proposed. In [BBD14a], analytical expressions for the outage probability and average content delivery rate in a network of SBSs equipped with caches are derived. Ref. [PT13] proposes an approximation algorithm for distributed coded caching, where the goal is to minimize the probability that mobile users get parts of content delivered from the MBS instead of the SBSs. A small cell network in which the MBS and the SBSs can perform multicast transmissions is considered in [PIST16] and a multicast-aware caching scheme is presented for minimizing the energy consumption.

However, in reality, prior knowledge about content popularity may not be available at a local cache. Therefore, a second line of literature investigates cache content place-

ment *under missing knowledge about content popularity*. In this case, the controller of a local cache may apply a worst-case approach by trying to optimize cache content with respect to a worst-case request scenario. For instance, Ref. [MAN14] combines caching at user devices with a coded multicast transmission in the delivery phase and derives information-theoretic gains of caching under a worst-case approach since content popularity is assumed to be unknown. The proposed coded caching approach is optimal up to a constant factor.

As an alternative to such a worst-case approach, the controller of the local cache may take a machine-learning-based approach and *learn* the content popularity distribution. Refs. [BBD14b, BBZ+15] declare a *proactive caching paradigm* and propose a proactive caching algorithm for small cell networks. Using collaborative filtering, a fixed global content popularity distribution is estimated in a training phase based on a given training set of content popularities. The learned content popularity distribution is then used for selecting the cache content to maximize the average user request satisfaction ratio based on the users' required delivery rates. A MAB framework is used in [BG14b], where an SBS learns a fixed content popularity distribution online by regularly updating its cache content and observing the numbers of requests for the cached files. Over time, the SBS thus optimizes its cache content placement to maximize the traffic served locally. The proposed framework is extended in [BG14c, BG14a] for a wireless infostation by additionally taking into account the costs for adding files to the cache and by deriving theoretical sublinear upper regret bounds for the proposed algorithms. Ref. [SAT+14] proposes another extension of the above MAB framework, which incorporates coded caching and hence exploits the topology of users' connections to the SBSs. In [EBSLa14], using a spectral clustering algorithm, users are clustered into groups of similar interests based on their requests in a training phase. Then, each user group is assigned to one SBS. Each SBS then learns the content popularity of its fixed user group over time. Therefore, the approach in [EBSLa14] relies on a stable user population.

The above discussed related work on learning-based cache content placement can be categorized as follows. Learning approaches are either tailored to a specific type of content popularity distribution [SAT+14], such that the approach is limited to cases where the type of distribution is known a priori, or they are *model-free* [BBD14b, BBZ+15, BG14b, BG14c, BG14a, SAT+14, EBSLa14] in the sense that they work for any type of content popularity distribution. Moreover, learning approaches are either based on *offline* learning [BBD14b, BBZ+15], or on *online* learning [BG14b, BG14c, BG14a, SAT+14, EBSLa14], the latter ones being able to better adapt to varying content popularities. Some learning approaches require a *training phase* [BBD14b, BBZ+15, EBSLa14], while others manage to only learn during run

Table 4.1. Related work on learning-based cache content placement with placement and delivery phase and detailed comparison with proposed algorithm.

Reference	[BBD14b], [BBZ+15]	[BG14b], [BG14c], [BG14a]	[SAT+14]	[EBSLa14]	This work
Model-free	Yes	Yes	No	Yes	Yes
Type of learning	Offline	Online	Online	Online	Online
Free of training phase	No	Yes	Yes	No	Yes
Regret bounds	No	Yes	No	No	Yes
Diversity in content popularity	No	No	No	Yes	Yes
User context-aware	No	No	No	No	Yes
Service differentiation	No	No	No	No	Yes

time [BG14b, BG14c, BG14a, SAT+14]. While some algorithms are only numerically evaluated [BBD14b, BBZ+15, SAT+14, EBSLa14], other related works are able to give performance guarantees by providing analytical *regret bounds* for their learning algorithms [BG14b, BG14c, BG14a]. Additionally, while most related works assume that there exists one global popularity distribution and that all user requests follow this distribution [BBD14b, BBZ+15, BG14b, BG14c, BG14a, SAT+14], only few works take into account that there can be *diversity in content popularity across the user population* since different users may favor different content [EBSLa14]. Among the related works, none takes into account that the users' content preferences may depend on their *contexts*, which is needed for proactive cache content placement in order to adapt to the preferences of mobile users with different contexts. Moreover, none of the literature takes into account that cache content placement should reflect the cache operator's specific objective who may want to offer *service differentiation* to its customers.

Table 4.1 gives an overview of the discussed related work on learning-based cache content placement with a placement and a delivery phase and provides a detailed comparison of the related work with the proposed algorithm. In contrast to the related work, cf. Table 4.1, we propose a context-aware proactive caching algorithm, which for

the first time *jointly* considers the following aspects:

(i) The proposed algorithm does not assume a priori knowledge about content popularity, which might be externally given or estimated in a separate training phase. Instead, *without requiring a training phase* and *model-free*, the proposed algorithm learns content popularity *online* by regularly updating the cache content and observing the users' requests for cache content. Based on an analytical bound, we give *performance guarantees* of the proposed algorithm, and prove that the learned cache content placement strategy converges to the optimal cache content placement strategy which maximizes the expected number of cache hits.

(ii) The proposed algorithm explicitly allows different content to be favored by different users. Incorporating such *diversity in content popularity across the user population* makes the proposed algorithm suitable for mobile scenarios, where users with different preferences connect to the wireless caching entity over time.

(iii) The underlying model explicitly incorporates that content popularity depends on a user's context, such as her/his personal characteristics, equipment, or external factors, and the proposed *context-aware* caching algorithm learns this context-specific content popularity. Using the proposed algorithm, the controller of a local cache may proactively cache content specifically tailored to the preferences of currently connected users based on what it has previously learned in similar situations, instead of simply caching those files that are popular on average, across the entire population of users.

(iv) The proposed algorithm takes into account the operator's specific objective by allowing for *service differentiation*.

4.3 System Model

4.3.1 Introduction

In this section, we propose a model for context-aware proactive caching in a local cache at the edge of the wireless network. In accordance with Section 2.2.1, the proposed overall model consists of the following five components:

(i) A *network model* is formulated specifying the assumptions on the wireless network and introducing the *wireless local caching entity* that performs caching at the

edge. The network model is applicable to different types of caching entities, such as SBSs equipped with caches, or wireless infostations.

(ii) A *context model* is defined, which includes side information that may impact a user's content preferences and hence the content popularity in a local cache.

(iii) As general performance criterion to be maximized, the number of cache hits is considered. However, if the cache operator wishes to offer service differentiation, the performance criterion to be maximized is the number of weighted cache hits, where different cache hits may have different weights. Hence, two submodels with respect to the performance criterion are formulated.

  (a) Since the number of cache hits depends on the content popularity, which in turn depends on the users' preferences and hence on the users' contexts, a *model of context-specific content popularity* is formulated, which explicitly allows different content to be favored by different users and that content popularity depends on a user's context.

  (b) A *model of service differentiation* is formulated, which allows the cache operator to provide differentiated services to its customers.

(iv) A decentralized *architecture of decision making* is proposed, where caching decisions are taken by the wireless local caching entity.

(v) An *action model* is formulated, which determines the different choices of the wireless local caching entity, namely, which content should and which should not be cached.

Based on the above, we also design an internal system architecture of the wireless local caching entity for context-aware proactive caching.

4.3.2 Network Model

We consider a *wireless local caching entity* which may, for example, be an SBS equipped with a cache in a small cell network or a wireless infostation [GBMY97, IR02, BG14c, BG14a]. The caching entity has a limited storage capacity and a reliable backhaul link to the core network. In its cache memory, the caching entity may store up to m files from a finite file library $\mathcal{F}$, which consists of $|\mathcal{F}| \in \mathbb{N}$ files, where we assume that all files are of the same size. This assumption can be made without loss of generality since otherwise, a finer packetization could be used and files could be divided into

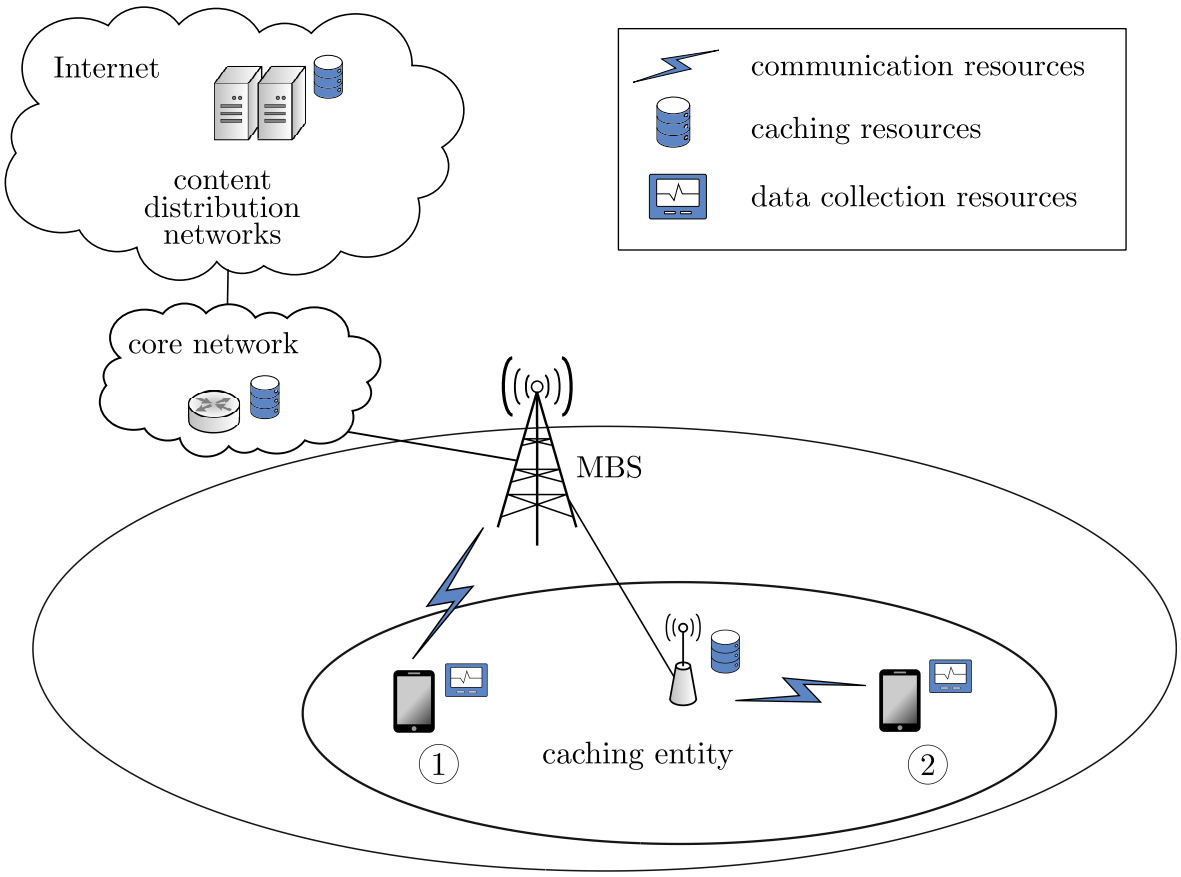


Figure 4.1. Network model.

smaller blocks of the same size [SGD+13,LXvdSL16b]. An illustration of the considered network model is depicted in Figure 4.1 and will be discussed in detail further below. We assume that the caching system operates in discrete time slots $t = 1, 2, ..., T$, where T denotes the finite time horizon. Users located in the coverage area may connect to the caching entity in order to request locally cached files. Due to the users' mobility, the set of connected users may change dynamically over time. We assume that at most $U_{\max} \in \mathbb{N}$ users may be simultaneously connected to the caching entity. The number of users connected to the caching entity in time slot t is denoted by U_t, with $1 \leq U_t \leq U_{\max}$, where we assume that the set of users does not change *within* a time slot. However, the model explicitly allows that the set of connected users changes from one time slot to another, and hence takes user mobility into account. We do not make any assumptions on the nature of the arrival process of the number of users other than that the sequence $\{U_t\}_{t=1,...,T}$ is not influenced by caching decisions taken by the caching entity.

In each time slot, the caching entity broadcasts the information about the current cache content in order to inform the connected users about it [BG14b, BG14c, BG14a].

Whenever a user is interested in a file that is currently stored in the cache of the caching entity, the user's device requests the file directly from the caching entity. Upon receiving a request for one of its cached contents, the caching entity serves the corresponding user via local communication. In this case, there is no additional traffic induced on neither the macro cellular network nor the backhaul network. Whenever a user requests a file that is not cached in the caching entity, the user's device does not request the file from the caching entity, but instead, it connects to an MBS to receive the file via the macro cellular network. The MBS then downloads the file from the core network or from a content distribution network via its backhaul connection. Hence, in this case, traffic occurs on both the macro cellular as well as the backhaul network.

To sum up, the caching entity may only observe *cache hits*, i.e., requests for cached files, but it cannot observe *cache misses*, i.e., requests for non-cached files, since the latter are directly handled via the macro cellular network. While this restriction is usually not used in wired caching scenarios, it is reasonable for wireless caching since it prevents the caching entity from being congested by cache misses [BG14b, BG14c, BG14a]. On the contrary, learning content popularity is more difficult under this restriction. In the illustration of the considered network model in Figure 4.1, device 1 connects to the MBS to request a file since the desired file is not locally cached in the caching entity. This corresponds to a cache miss from the point of view of the caching entity. Device 2 requests a cached file from the caching entity. This corresponds to a cache hit at the caching entity.

4.3.3 Context Model

Cache content placement requires knowledge about content popularity. Since different users may favor different content, content popularity may vary across the user population. Moreover, various factors may impact a user's content preferences. Such factors may be summarized under the term *context*. Some examples of context dimensions are presented in Table 4.2. Among relevant context dimensions are *personal characteristics*, such as demographic factors (e.g., age [MS10], gender [HL05]), personality [RGZ11], or mood [Zil88]. Other context dimensions that may influence a user's content preferences are the characteristics of her/his *user equipment* [ZGC+14], such as the type of device used to access and consume the content (e.g., smart phone, tablet), its capabilities, or its battery status. Moreover, *external factors* may affect a user's content preferences, such as the user's location, the time of day, the day of the week, and the taking place of events (e.g., soccer match, concert). This categorization only gives examples of possible context dimensions, but obviously the categorization

Table 4.2. Examples of context dimensions of different types.

Type	**Examples**
Personal characteristics	Demographic factors, personality, mood
User equipment	Type of device, device capabilities, battery status
External factors	Location, time of day, day of the week, events

is neither exhaustive nor is knowledge about the impact of the context dimensions on content popularity available. Moreover, while a caching entity may have access to user context by exploiting the data collection resources of mobile devices, as indicated in Figure 4.1, or information from external sources (e.g., social media platforms), this access may be restricted to only *certain* context dimensions due to reasons of privacy or communication overhead. However, the proposed model and algorithm *do not* rely on *specific* context dimensions to be available. Instead, the model and algorithm can use the information that *is* collected from the user. Hence, if context information may be accessed by the caching entity, the proposed algorithm may exploit this available context information.

Formally, the context information which may be accessed by the caching entity is modeled as follows. We denote the number of monitored context dimensions per user by D and we denote the D-dimensional context space by $\mathcal{X}$. The context space $\mathcal{X}$ is assumed to be bounded and can hence be set to $\mathcal{X} := [0,1]^D$ without loss of generality. Hence, we assume that the context of a user is described in terms of D context dimensions, and that in each of the D context dimensions, the user context is described by a value from $[0,1]$, such that the overall context of a user is hence a vector in $[0,1]^D$. Clearly, in practice, a pre-processing might be needed here which maps the actual monitored user context information to a vector in $\mathcal{X} = [0,1]^D$.

In time slot t, the context vector of a currently connected user $i \in \{1,...,U_t\}$ is denoted by $\mathbf{x}_{t,i} \in \mathcal{X}$. Moreover, the set of context vectors of all connected users in time slot t is denoted by $\mathcal{X}_t := \{\mathbf{x}_{t,i}\}_{i=1,...,U_t}$. We do not make any assumptions on the nature of the context arrival process other than that the sequence $\{\mathcal{X}_t\}_{t=1,...,T}$ is not influenced by caching decisions taken by the caching entity.

4.3.4 Model of Context-Specific Content Popularity

Next, a model of content popularity in dependence of a user's context is given. As stated before, we assume that the caching system operates in discrete time slots. A user may request several files within one time slot. The number of times a user with context vector $\mathbf{x} \in \mathcal{X}$ requests a file $f \in \mathcal{F}$ within one time slot is a random variable with unknown distribution. This random demand is denoted by $d_f(\mathbf{x})$ and its expected value is denoted by $\mu_f(\mathbf{x}) := \mathbb{E}[d_f(\mathbf{x})]$. The random demand $d_f(\mathbf{x})$ is assumed to take values in $[0, R_{\max}]$, where $R_{\max} \in \mathbb{N}$ is the maximum possible number of requests for the same file a user may submit within one time slot. Hence, if $R_{\max} > 1$, it is possible for a user to request the same file repeatedly within one time slot.

Based on this notation, in time slot t, given the context vector $\mathbf{x}_{t,i}$ of a connected user $i \in \{1, ..., U_t\}$, the random variable describing the demand for a file $f \in \mathcal{F}$ of that user is given by $d_f(\mathbf{x}_{t,i})$ and its expected value is given by $\mu_f(\mathbf{x}_{t,i})$. We assume that in time slot t, the random variables $\{d_f(\mathbf{x}_{t,i})\}_{i=1,..,U_t, f\in\mathcal{F}}$ are independent of each other and each random variable $d_f(\mathbf{x}_{t,i})$ is independent of past caching decisions and previous demands. Moreover, by $d_f(\mathbf{x}_{t,i}, t)$, we denote the actual instantaneous demand of user $i \in \{1, ..., U_t\}$ in time slot t, i.e., the realization of the random variable $d_f(\mathbf{x}_{t,i})$ in time slot t.

4.3.5 Model of Service Differentiation

The general goal of the caching entity is to minimize the load on the macro cellular network and the backhaul network by selecting the cache content in such a way that the traffic it can serve locally is maximized, which corresponds to maximizing the number of local cache hits.

However, the caching entity may additionally need to take into account the cache operator's specific objective since in certain cases, a cache operator may want to offer service differentiation to its customers. Service differentiation might not be adequate for an MNO operating an SBS since due to net neutrality restrictions, the MNO should actually purely maximize the number of cache hits. However, service differentiation may be interesting for the operator of an infostation, e.g., a content provider or a third party operator, whose customers may be both users and content providers.

As an example, the operator of an infostation may prioritize certain users by caching content favored by these users. Hence, a cache hit produced by a prioritized user corresponds to a higher value for the operator, as compared to a cache hit by a regular user.

To formalize such requirements for service differentiation among users, we consider that different service groups exist and that each user is member of one service group. Formally, we consider a finite set $\mathcal{G}$ of service groups. For service group $g \in \mathcal{G}$, let $v_g \geq 1$ denote a fixed and known weight associated with receiving one cache hit from a user of service group g. Let $v_{\max} := \max_{g \in \mathcal{G}} v_g$. There are different approaches how the operator may choose the weights associated with the different service groups. For example, the weights may be chosen based on payments, e.g., users may buy higher weights. As another example, the weights may be chosen based on subscription, e.g., subscribers may obtain priority compared to one-time users. Also, the weights may be chosen based on the importance of users for the operator, e.g., in case the operator wants to prioritize certain target groups due to advertisement or reputation. Alternatively, the weights may even be chosen by the users themselves based on their interests, e.g., users may indicate their degree of openness in exploring other than their most preferred content. In any case, cache content placement that takes into account such service weights should aim at maximizing the number of *weighted* cache hits. If content popularity is heterogeneous across the user population, by maximizing the number of weighted cache hits, service differentiation affects the selection of cache content.

The case discussed above is not the only type of service differentiation that may be interesting for the operator of an infostation. Especially if the operator is a third party whose customers are different content providers, the operator may want to provide differentiated services to content providers. For example, the operator may prioritize certain content providers by preferably caching their content. To formalize such requirements for service differentiation among different content providers, we consider that each content is associated with a weight. Formally, for file $f \in \mathcal{F}$, we consider a fixed and known prioritization weight $w_f \geq 1$ associated with receiving one cache hit for file f. Let $w_{\max} := \max_{f \in \mathcal{F}} w_f$. The prioritization weights may either be chosen individually for each file or per content provider.

The case that no service differentiation is needed and hence the goal remains to maximize the number of (non-weighted) cache hits, may formally be modeled as the case where there is only one service group g with weight $v_g = 1$ and the prioritization weights satisfy $w_f = 1$ for all $f \in \mathcal{F}$. Therefore, the case without service differentiation is a special case of the proposed model and while this special case is not treated explicitly in the sequel, it is implicitly contained in the remainder of this chapter.

In time slot t, the service group to which a currently connected user $i \in \{1, ..., U_t\}$ belongs is denoted by $g_{t,i} \in \mathcal{G}$ and its corresponding weight is given by $v_{g_{t,i}}$. Moreover, the set of service groups to which the set of users in time slot t belongs is denoted by $\mathcal{G}_t := \{g_{t,i}\}_{i=1,...,U_t}$. We do not make any assumptions on the nature of the arrival

process of the service groups other than that the sequence $\{\mathcal{G}_t\}_{t=1,...,T}$ is not influenced by caching decisions taken by the caching entity.

4.3.6 Architecture of Decision Making

We propose to use a decentralized architecture of decision making, cf. Section 2.2.3, in which the wireless local caching entity takes local caching decisions. We use a decentralized architecture since the unknown local content popularity is not necessarily the same as the global content popularity [GALM07, ZSGK09, BSW12] and since the set of mobile users with potentially different interests in the vicinity of a caching entity changes over time.

4.3.7 Action Model

The caching entity's actions are formalized as follows. We introduce a binary variable $y_{t,f}$ for each file $f \in \mathcal{F}$ and each time slot $t \in \{1, ..., T\}$, where

$$y_{t,f} := \begin{cases} 1, & \text{if file } f \text{ is available in the cache in time slot } t, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Since the caching entity may store up to m files from the finite library $\mathcal{F}$ in its local cache, the following capacity constraints

$$\sum_{f \in \mathcal{F}} y_{t,f} \leq m, \quad t = 1, ..., T, \tag{4.2}$$

have to hold. We denote the set of cached files in time slot t by

$$\mathcal{C}_t := \{f \in \mathcal{F} : y_{t,f} = 1\}. \tag{4.3}$$

4.3.8 Internal Architecture of Wireless Local Caching Entity

The internal system architecture of the wireless local caching entity for context-aware proactive caching is designed similarly as the architecture presented in [LXvdSL16b, LXvdSL16a]. Figure 4.2 shows an illustration of the context-aware proactive caching architecture. The main building blocks are a *Local Cache*, a *Cache Management* entity, a *Learning Module*, a *Storage Interface*, a *User Interface*, and a *Context Monitor*. The

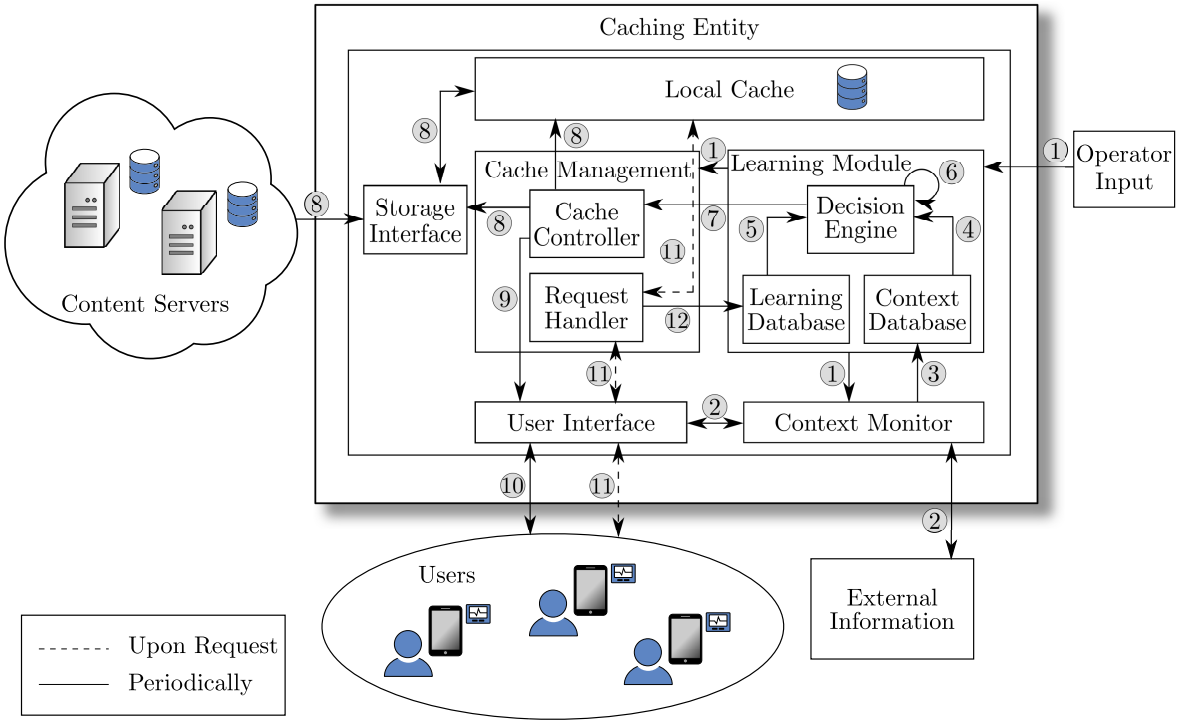


Figure 4.2. Context-aware proactive caching architecture.

Cache Management contains a *Cache Controller* and a *Request Handler*. The Learning Module consists of a *Decision Engine*, a *Learning Database*, and a *Context Database*. The workflow comprises several phases as enumerated in Figure 4.2 and is described in the sequel.

- Initialization
  (1) The cache operator informs the Learning Module about the goal of caching (including details about weights in case of service differentiation) and about the time horizon. Thereupon, the Learning Module fixes the appropriate periodicity of context monitoring and cache refreshment. Then, the Cache Management and the Context Monitor are informed about the periodicity.

- Periodic Context Monitoring and Cache Refreshment
  (2) The Context Monitor periodically retrieves context information about currently connected users from the User Interface and optionally collects additional context information from external sources (e.g., social media platforms). In case of service differentiation, if different service groups exist, the Context Monitor also retrieves the service groups of the connected users. (3) The collected information is delivered from the Context Monitor to the Context Database in the Learning Module. (4) The Decision Engine periodically extracts the new context

information from the Context Database. (5) Based on a comparison with results stored in the Learning Database in previous time slots, (6) the Decision Engine decides which files to cache in the upcoming time slot. (7) The Cache Controller is instructed by the Decision Engine to refresh the cache content accordingly. (8) The Cache Controller compares the current and the required cache content and then removes non-required content from the cache. If required content is missing, the Cache Controller requests the Storage Interface to fetch the content from storage servers and to store it into the local cache. (9) Then, the User Interface is informed about the new cache content by the Cache Controller. (10) The User Interface informs currently connected users about the new cache content.

- User Requests
  (11) If a user requests a cached file, the User Interface forwards the request to the Request Handler. The Request Handler stores the request information, retrieves the file from the local cache and serves the user.

- Periodic Learning
  (12) Upon completion of a time slot, the Request Handler hands the information about all requests that have arrived in this time slot to the Learning Module. The Learning Module updates the Learning Database with the context information collected in the beginning of the time slot and with the number of requests for cached files that have arrived in the time slot according to the information from the Request Handler.

4.4 Problem Formulation

4.4.1 Formal Problem Statement

In this section, using the models from Section 4.3, we formulate the problem of context-aware proactive caching at the edge for maximizing the number of cache hits under missing knowledge about content popularity to be locally solved by a wireless local caching entity. As stated before, the caching system operates in discrete time slots $t = 1, 2, ..., T$, where T denotes the finite time horizon. As depicted in Figure 4.3, the following sequence of operations is executed in each time slot t:

(i) The caching entity monitors the contexts $\mathcal{X}_t = \{\mathbf{x}_{t,i}\}_{i=1,...,U_t}$ of the U_t currently connected users as well as the service groups $\mathcal{G}_t = \{g_{t,i}\}_{i=1,...,U_t}$ to which the users belong.

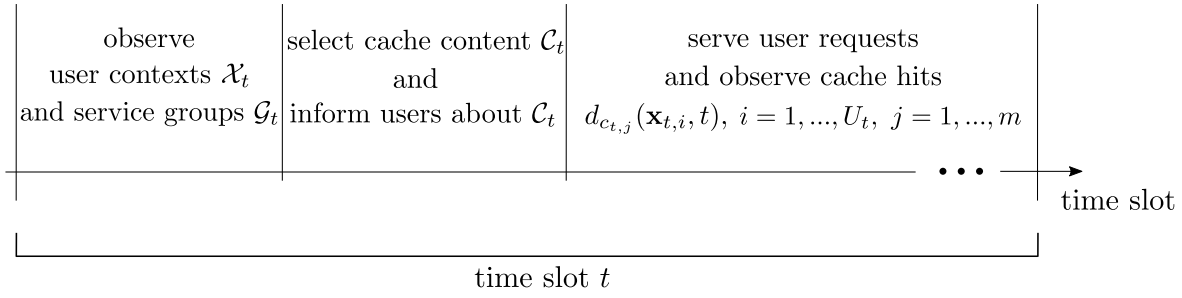


Figure 4.3. Sequence of operations of context-aware proactive caching in time slot t.

(ii) The caching entity updates the cache content based on the contexts $\mathcal{X}_t$, the service groups $\mathcal{G}_t$ and their corresponding service weights, the file prioritization weights w_f, $f \in \mathcal{F}$, and knowledge from previous time slots. Then, the caching entity informs the connected users about the current cache content $\mathcal{C}_t = \{c_{t,1}, ..., c_{t,m}\}$.

(iii) Until the end of time slot t, users may request the currently cached files as given in $\mathcal{C}_t$. The caching entity serves the users' requests for cached files. Moreover, the caching entity observes the demand $d_{c_{t,j}}(\mathbf{x}_{t,i}, t)$ of each user $i = 1, ..., U_t$ for each cached file $c_{t,j} \in \mathcal{C}_t$, $j = 1, ..., m$, during this time slot, i.e., the caching entity monitors the number of cache hits for each cached file.

Consider a *sequence of T time slots with arbitrary user arrivals*, i.e., consider a sequence $t = 1, ..., T$ with an arbitrary sequence of user numbers $\{U_t\}_{t=1,...,T}$ and with arbitrary sequences of user contexts $\{\mathcal{X}_t = \{\mathbf{x}_{t,i}\}_{i=1,...,U_t}\}_{t=1,...,T}$ and service groups $\{\mathcal{G}_t = \{g_{t,i}\}_{i=1,...,U_t}\}_{t=1,...,T}$. The goal of the caching entity is to select the cache content in such a way that the expected cumulative number of weighted cache hits up to the finite time horizon T is maximized. Based on the action model in Section 4.3.7 and the capacity constraints in (4.2), the problem of cache content placement can be formally written as

$$\begin{aligned} \max \ & \sum_{t=1}^{T} \sum_{f \in \mathcal{F}} y_{t,f} w_f \sum_{i=1}^{U_t} v_{g_{t,i}} \mu_f(\mathbf{x}_{t,i}) \\ \text{s.t.} \ & \sum_{f \in \mathcal{F}} y_{t,f} \leq m, \quad t = 1, ..., T, \\ & y_{t,f} \in \{0, 1\}, \quad f \in \mathcal{F},\ t = 1, ..., T, \end{aligned} \tag{4.4}$$

with $y_{t,f}$ of (4.1), w_f and $v_{g_{t,i}}$ as defined in Section 4.3.5, $\mu_f(\mathbf{x}_{t,i})$ as defined in Section 4.3.4 and the constraints from (4.2).

Problem (4.4) depends on the context-specific content popularity, i.e., the expected demands of connected users as a function of their contexts.

4.4.2 Oracle Solution

First, we classify Problem (4.4) under the assumption that the caching entity *had* a priori knowledge about context-specific content popularity. Hence, only in this section, suppose that the caching entity *had* a priori knowledge about context-specific content popularity *like an omniscient oracle*, i.e., assume that the caching entity would know the expected demand $\mu_f(\mathbf{x}) = \mathbb{E}\left[d_f(\mathbf{x})\right]$ for any pair consisting of a context vector $\mathbf{x} \in \mathcal{X}$ and a file $f \in \mathcal{F}$. Under this assumption, Problem (4.4) corresponds to an ILP problem, cf. Section 2.3.2.2. Since the sub-problems for the different time slots are not coupled, Problem (4.4) can be decoupled into T independent sub-problems.

The sub-problem associated to any time slot t corresponds to a knapsack problem, see Section 2.3.2.3, with a knapsack of capacity m and with $|\mathcal{F}|$ items, where item $f \in \mathcal{F}$ has a unit weight and a non-negative profit $w_f \sum_{i=1}^{U_t} v_{g_{t,i}} \mu_f(\mathbf{x}_{t,i})$. Due to the unit weights, each sub-problem is actually a special case of the knapsack problem which may be solved efficiently. In detail, the optimal solution of the sub-problem in time slot t can be easily computed in a running time of $O(|\mathcal{F}| \log(|\mathcal{F}|))$ as follows. Given the contexts $\mathcal{X}_t$ and the service groups $\mathcal{G}_t$, the optimal solution is given by ranking the files in $\mathcal{F}$ according to their expected weighted demands and by caching the m highest ranked files. We denote these *top-m files for pair ($\mathcal{X}_t$, $\mathcal{G}_t$)* by $f_1^*(\mathcal{X}_t, \mathcal{G}_t), f_2^*(\mathcal{X}_t, \mathcal{G}_t), ..., f_m^*(\mathcal{X}_t, \mathcal{G}_t) \in \mathcal{F}$. Formally, for $j = 1, ..., m$, they satisfy

$$f_j^*(\mathcal{X}_t, \mathcal{G}_t) \in \underset{f \in \mathcal{F} \setminus (\bigcup_{k=1}^{j-1} \{f_k^*(\mathcal{X}_t, \mathcal{G}_t)\})}{\operatorname{argmax}} w_f \sum_{i=1}^{U_t} v_{g_{t,i}} \mu_f(\mathbf{x}_{t,i}), \tag{4.5}$$

where $\bigcup_{k=1}^{0} \{f_k^*(\mathcal{X}_t, \mathcal{G}_t)\} := \emptyset$. Note that several files may have the same expected demands, i.e., the optimal set of files may not be unique, which is also captured here. Moreover, by $\mathcal{C}_t^*(\mathcal{X}_t, \mathcal{G}_t) := \bigcup_{k=1}^{m} \{f_k^*(\mathcal{X}_t, \mathcal{G}_t)\}$, we denote an optimal choice of files to cache in time slot t. Then, an optimal overall solution to Problem (4.4) is given by the collection

$$\mathcal{C}^* := \{\mathcal{C}_t^*(\mathcal{X}_t, \mathcal{G}_t)\}_{t=1,...,T}. \tag{4.6}$$

We call the optimal collection in (4.6) the *oracle solution* since it may be calculated by an omniscient oracle based on a priori knowledge about context-specific content popularity.

4.4.3 Contextual Multi-Armed Bandit Formulation

Now, we characterize Problem (4.4) under missing knowledge about content popularity since typically, the caching entity *does not* have a priori knowledge about content pop-

ularity. In this case, the caching entity cannot simply solve Problem (4.4) as described in Section 4.4.2 since it does not know the expected demands $\mu_f(\mathbf{x}) = \mathbb{E}[d_f(\mathbf{x})]$. Hence, a *machine-learning-based* approach for designing the decision agent, cf. Section 2.3.1, is required since the caching entity can only learn the content popularity by caching different files over time and subsequently observing their cache hits. Considering the problem formulation in Section 4.4.1, Problem (4.4) can be understood as a contextual MAB problem, cf. Section 2.3.3.4, as follows. The caching entity corresponds to an agent which needs to sequentially select from a set of actions. In our case, the set of actions is given by the set $\mathcal{F}$ of files. There is a sequence of time slots $t = 1, ..., T$, or, rounds, in the wording of Section 2.3.3.4, each of them consisting of three events happening sequentially. In each time slot, the caching entity first observes user contexts $\mathcal{X}_t = \{\mathbf{x}_{t,i}\}_{i=1,...,U_t}$ and service groups $\mathcal{G}_t = \{g_{t,i}\}_{i=1,...,U_t}$. This corresponds to a set of contexts revealed to an agent in the beginning of a round. Then, the caching entity selects a subset of m files from set $\mathcal{F}$. This corresponds to an agent selecting a subset of actions. Then, the caching entity observes the demands for each cached file. This corresponds to an agent receiving a reward for each selected action. Taking into account the assumptions about the arrival processes of the number of users, their contexts and service groups in Sections 4.3.2 – 4.3.4, Problem (4.4) hence corresponds to a contextual MAB problem with a similar model as the one presented in Section 2.3.3.4.

The main difference between Problem (4.4) and the model in Section 2.3.3.4 is that in Problem (4.4), the agent may select several (in detail, m) actions per round instead of only one as in the model in Section 2.3.3.4. Hence, Problem (4.4) may formally be called a contextual *combinatorial* MAB problem, cf. Section 2.3.3.2. However, neither the objective function nor the constraint in Problem (4.4) is combinatorial, in contrast to "real" combinatorial MAB problems, where the reward may depend on the subset of selected actions and constraints may be combinatorial. Therefore, Problem (4.4) is therefore more accurately a contextual MAB problem with several action selections per round, but not of combinatorial nature.

Having formulated Problem (4.4) as a contextual MAB problem, the task of the caching entity is as follows. The caching entity needs to learn the unknown expected demands $\mu_f(\mathbf{x}) = \mathbb{E}[d_f(\mathbf{x})]$ over time by regularly updating the cache content and observing the users' contexts and their demands for the cache content. Since the caching entity needs both to learn expected demands and to maximize the number of weighted cache hits online over time, the caching entity has to find a suitable trade-off between caching files about which little information is available (*exploration*) and files of which it believes that they will yield a high number of cache hits (*exploitation*). Which files to cache in a time slot depends on the history of cached files in the past and the corresponding observed demands. An algorithm which maps the history to the selections of

files to be cached is called a *learning algorithm*. The loss of learning can be evaluated by comparing the learning algorithm and the oracle solution given in (4.6) in terms of their respective achieved cumulative weighted numbers of cache hits. Formally, for a sequence of T time slots with arbitrary user arrivals, the *regret* of learning with respect to the oracle solution is given by

$$R(T) = \sum_{t=1}^{T} \sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \left(w_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)} \, \mathbb{E}\left[d_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)}(\mathbf{x}_{t,i}, t) \right] - \mathbb{E}\left[w_{c_{t,j}} d_{c_{t,j}}(\mathbf{x}_{t,i}, t) \right] \right), \quad (4.7)$$

where $d_{c_{t,j}}(\mathbf{x}_{t,i}, t)$ denotes the instantaneous demand for the cached file $c_{t,j} \in \mathcal{C}_t$ of user i with context vector $\mathbf{x}_{t,i}$ at time t. Here, the expectation is taken with respect to the selections made by the learning algorithm and the distributions of the demands.

4.5 Proposed Algorithm

Using the formulation as contextual MAB problem in Section 4.4.3, we propose an online learning algorithm for context-aware proactive caching that is based on the contextual MAB algorithms in [LPP10, Sli14, TvdS15a, TZvdS14], cf. Section 2.3.3.4 for a discussion of these works within the context of the MAB literature. The closest contextual MAB algorithm to our proposed algorithm is presented in [TZvdS14]. In their considered contextual MAB problem, there are several learners, each of which observes a single context arrival in each time slot. Then, each learner needs to select a subset of actions with the goal to maximize the sum of expected rewards. While multiple learners are present in [TZvdS14], only one learner is present in the caching problem considered in this chapter. Specifically, this learner corresponds to the caching entity which needs to select a subset of files to cache in each time slot. Moreover, the algorithm proposed below extends the algorithm from [TZvdS14] as follows:

- The proposed algorithm allows multiple contexts to arrive in each time slot, and the algorithm then selects a subset of actions with the goal to maximize the sum of expected rewards given the set of context arrivals. In the considered caching problem, where each user has her/his own context, this allows the caching entity to observe the set of contexts of the set of currently connected users and then to cache a subset of files with the goal to maximize the sum of expected numbers of cache hits given the users' contexts.
- Moreover, the proposed algorithm allows each arriving context to be annotated with a weight, so that when different contexts arrive within the same time slot,

differentiated services may be provided per context, by selecting a subset of actions with the goal to maximize the sum of expected weighted rewards. In the considered caching problem, this allows the caching entity to prioritize certain users when selecting the cache content, by placing more weight on expected cache hits of prioritized users.

- Finally, the proposed algorithm allows each action to be annotated with a weight, such that certain actions may be prioritized. In the considered caching problem, this allows the caching entity to prioritize certain files when selecting the cache content, by placing more weight on these files.

Here, we give a short overview of the proposed algorithm. The proposed algorithm is based on the assumption that users with similar context on average have similar content preferences. Under this natural assumption, the algorithm can exploit observations of the users' context information together with the users' requests for cached files for improving future caching decisions. During initialization, the algorithm uniformly partitions the context space, i.e., it splits the context space uniformly into smaller sets of similar contexts. This partition of the context space is later used at run time, where the algorithm learns content popularity independently in each of these sets of similar contexts. At run time, the algorithm operates in discrete time slots. In the beginning of a time slot, the algorithm observes the contexts of currently connected users. Then, the algorithm selects the files to cache in this time slot. For this purpose, the algorithm first checks based on a control function, which will be defined below, if there are files that have not been cached sufficiently often before. On the one hand, if this is the case, the algorithm enters an exploration phase and caches a random set of files from the set of files that have not been cached sufficiently often before. Exploration phases enable the algorithm to learn the popularity of files that have not been cached often before. On the other hand, if this is not the case, the algorithm enters an exploitation phase and caches files that on average were requested most when cached in previous time slots with similar user contexts. After the selected files have been cached, the algorithm observes the users' requests for these files until the end of the time slot. In this way, the algorithm learns context-specific content popularity over time.

We call the proposed algorithm the *context-aware proactive caching* (CAC) algorithm. The pseudocode of CAC is given in Algorithm 4.1. In detail, CAC proceeds as follows. During initialization, CAC creates a uniform partition $\mathcal{P}_T$ of the context space $\mathcal{X} = [0,1]^D$ into $(h_T)^D$ D-dimensional hypercubes of identical size $\frac{1}{h_T} \times \ldots \times \frac{1}{h_T}$. The parameter h_T is an input to the algorithm which determines the number of sets in the partition. An adequate choice of the parameter h_T will be proposed in Section 4.6.1.

Algorithm 4.1 CAC: Context-Aware Proactive Caching Algorithm

Input: T, h_T, $K : \{1, ..., T\} \to \mathbb{R}_+$
Initialize context partition: Create partition $\mathcal{P}_T$ of context space $[0,1]^D$ into $(h_T)^D$ hypercubes of identical size
Initialize counters: For all $f \in \mathcal{F}$ and all $p \in \mathcal{P}_T$, set $N_{f,p} = 0$
Initialize estimated demands: For all $f \in \mathcal{F}$ and all $p \in \mathcal{P}_T$, set $\hat{\mu}_{f,p} = 0$
for each $t = 1, ..., T$ **do**
    Observe number U_t of currently connected users
    Observe user contexts $\mathcal{X}_t = \{\mathbf{x}_{t,i}\}_{i=1,...,U_t}$ and service groups $\mathcal{G}_t = \{g_{t,i}\}_{i=1,...,U_t}$
    Find $\mathcal{P}_t = \{p_{t,i}\}_{i=1,...,U_t}$ such that $\mathbf{x}_{t,i} \in p_{t,i} \in \mathcal{P}_T, i = 1, ..., U_t$
    Compute the set of under-explored files $\mathcal{F}_t^{\text{ue}}$ in (4.8)
    if $\mathcal{F}_t^{\text{ue}} \neq \emptyset$ **then** ▷ Exploration
        $F_{\text{ue},t} = \text{size}(\mathcal{F}_t^{\text{ue}})$
        if $F_{\text{ue},t} \geq m$ **then**
            Select $c_{t,1}, ..., c_{t,m}$ randomly from $\mathcal{F}_t^{\text{ue}}$
        else
            Select $c_{t,1}, ..., c_{t,F_{\text{ue},t}}$ as the $F_{\text{ue},t}$ files from $\mathcal{F}_t^{\text{ue}}$
            Select $c_{t,F_{\text{ue},t}+1}, ..., c_{t,m}$ as the $(m - F_{\text{ue},t})$ files $\hat{f}_{1,\mathcal{P}_t,\mathcal{G}_t}(t), ..., \hat{f}_{m-F_{\text{ue},t},\mathcal{P}_t,\mathcal{G}_t}(t)$ from (4.9)
        end if
    else ▷ Exploitation
        Select $c_{t,1}, ..., c_{t,m}$ as the m files $\hat{f}_{1,\mathcal{P}_t,\mathcal{G}_t}(t), ..., \hat{f}_{m,\mathcal{P}_t,\mathcal{G}_t}(t)$ from (4.10)
    end if
    Broadcast information about cache content $\mathcal{C}_t$
    Observe demand $d_{j,i}$ of each user $i = 1, ..., U_t$ for each file $c_{t,j}, j = 1, ..., m$
    for $i = 1, ..., U_t$ **do**
        for $j = 1, ..., m$ **do**
            $\hat{\mu}_{c_{t,j},p_{t,i}} = \frac{\hat{\mu}_{c_{t,j},p_{t,i}} N_{c_{t,j},p_{t,i}} + d_{j,i}}{N_{c_{t,j},p_{t,i}} + 1}$ and $N_{c_{t,j},p_{t,i}} = N_{c_{t,j},p_{t,i}} + 1$
        end for
    end for
end for

Moreover, CAC initializes a counter $N_{f,p}(t)$ for each pair consisting of a file $f \in \mathcal{F}$ and a set $p \in \mathcal{P}_T$. The counter $N_{f,p}(t)$ corresponds to the number of times in which file $f \in \mathcal{F}$ was cached while a user with context from set p was connected to the caching entity before time slot t (i.e., if 2 users with context from set p were connected in one time slot and file f was cached, the counter is increased by 2). Additionally, CAC initializes the estimate $\hat{\mu}_{f,p}(t)$ of each pair consisting of a file $f \in \mathcal{F}$ and a set $p \in \mathcal{P}_T$. The estimate $\hat{\mu}_{f,p}(t)$ corresponds to the estimated demand for file $f \in \mathcal{F}$ under contexts from hypercube $p \in \mathcal{P}_T$ in time slot t. The estimated demand is calculated as the sample mean of previously observed demands: Let $\mathcal{E}_{f,p}(t)$ be the set of observed demands of users with context from set p when file f was cached before time slot t. If before time slot t, file f was never cached when users with context from set p appeared, we have $\mathcal{E}_{f,p}(t) = \emptyset$ and $\hat{\mu}_{f,p}(t) := 0$. Otherwise, the estimated demand of file f in set p is given by the sample mean $\hat{\mu}_{f,p}(t) := \frac{1}{|\mathcal{E}_{f,p}(t)|} \sum_{d \in \mathcal{E}_{f,p}(t)} d$. The set $\mathcal{E}_{f,p}(t)$ itself does not appear in Algorithm 4.1 since the estimated demand $\hat{\mu}_{f,p}(t)$ may be updated based on $\hat{\mu}_{f,p}(t-1)$, $N_{f,p}(t-1)$ and based on the observed demands in time slot $t-1$. Moreover, in Algorithm 4.1, the argument t is dropped from counters $N_{f,p}(t)$ and $\hat{\mu}_{f,p}(t)$ since previous values of these counters do not have to be stored.

In the beginning of each time slot t, CAC first observes the number U_t of currently connected users, the users' contexts $\mathcal{X}_t = \{\mathbf{x}_{t,i}\}_{i=1,\dots,U_t}$, and the service groups $\mathcal{G}_t = \{g_{t,i}\}_{i=1,\dots,U_t}$ to which the users belong. Then, CAC determines for each of the context vectors $\mathbf{x}_{t,i} \in \mathcal{X}_t$ the set $p_{t,i} \in \mathcal{P}_T$, to which the context vector belongs. Hence, for $\mathbf{x}_{t,i} \in \mathcal{X}_t$, CAC finds $p_{t,i} \in \mathcal{P}_T$ such that $\mathbf{x}_{t,i} \in p_{t,i}$ holds. The collection of these sets is denoted by $\mathcal{P}_t := \{p_{t,i}\}_{i=1,\dots,U_t}$. Then, the algorithm determines whether to enter an exploration phase or an exploitation phase. In order to determine which phase to enter, the algorithm checks whether there are files that have not been explored sufficiently often. For this purpose, the algorithm calculates the *set of under-explored files* $\mathcal{F}_t^{\text{ue}}$ based on

$$\mathcal{F}_t^{\text{ue}} := \cup_{i=1}^{U_t} \{f \in \mathcal{F} : N_{f,p_{t,i}}(t) \leq K(t)\}, \tag{4.8}$$

where $K : \{1, \dots, T\} \to \mathbb{R}_+$ is a deterministic, monotonically increasing control function, which is an input to the algorithm. The choice of the control function is crucial since it determines the trade-off between exploration and exploitation. An adequate choice of the control function, which guarantees a good balance in terms of this trade-off, will be proposed in Section 4.6.1.

Let $F_{\text{ue},t} := |\mathcal{F}_t^{\text{ue}}|$ be the size of set $\mathcal{F}_t^{\text{ue}}$. If the set $\mathcal{F}_t^{\text{ue}}$ is non-empty, i.e., $F_{\text{ue},t} > 0$, CAC enters an exploration phase. In case the set $\mathcal{F}_t^{\text{ue}}$ contains at least m elements, i.e., $F_{\text{ue},t} \geq m$, the algorithm randomly selects m files from $\mathcal{F}_t^{\text{ue}}$ to cache. In case the set $\mathcal{F}_t^{\text{ue}}$

contains less than m elements, i.e., $F_{\mathrm{ue},t} < m$, it selects all $F_{\mathrm{ue},t}$ files from $\mathcal{F}_t^{\mathrm{ue}}$ to cache. Since the cache is not fully filled by $F_{\mathrm{ue},t} < m$ files, $(m - F_{\mathrm{ue},t})$ additional files may be cached. In order to exploit knowledge obtained so far, CAC selects $(m - F_{\mathrm{ue},t})$ additional files from $\mathcal{F} \setminus \mathcal{F}_t^{\mathrm{ue}}$ as follows. CAC ranks the files in $\mathcal{F} \setminus \mathcal{F}_t^{\mathrm{ue}}$ according to their estimated weighted demands, and selects those $(m - F_{\mathrm{ue},t})$ files $\hat{f}_{1,\mathcal{P}_t,\mathcal{G}_t}(t), ..., \hat{f}_{m-F_{\mathrm{ue},t},\mathcal{P}_t,\mathcal{G}_t}(t) \in \mathcal{F} \setminus \mathcal{F}_t^{\mathrm{ue}}$ which satisfy for $j = 1, ..., m - F_{\mathrm{ue},t}$:

$$\hat{f}_{j,\mathcal{P}_t,\mathcal{G}_t}(t) \in \underset{f \in \mathcal{F} \setminus (\mathcal{F}_t^{\mathrm{ue}} \cup \bigcup_{k=1}^{j-1} \{\hat{f}_{k,\mathcal{P}_t,\mathcal{G}_t}(t)\})}{\operatorname{argmax}} w_f \sum_{i=1}^{U_t} v_{g_{t,i}} \hat{\mu}_{f,p_{t,i}}(t). \tag{4.9}$$

If the set of files defined by (4.9) is not unique, ties are broken arbitrarily. Note that by this procedure, even in exploration phases, the algorithm additionally exploits, whenever the number of under-explored files is smaller than the cache size.

If the set $\mathcal{F}_t^{\mathrm{ue}}$ is empty, CAC enters an exploitation phase and selects m files from $\mathcal{F}$ as follows. CAC ranks the files in $\mathcal{F}$ according to the estimated weighted demands, and selects those m files $\hat{f}_{1,\mathcal{P}_t,\mathcal{G}_t}(t), ..., \hat{f}_{m,\mathcal{P}_t,\mathcal{G}_t}(t) \in \mathcal{F}$ which satisfy for $j = 1, ..., m$:

$$\hat{f}_{j,\mathcal{P}_t,\mathcal{G}_t}(t) \in \underset{f \in \mathcal{F} \setminus \left(\bigcup_{k=1}^{j-1} \{\hat{f}_{k,\mathcal{P}_t,\mathcal{G}_t}(t)\}\right)}{\operatorname{argmax}} w_f \sum_{i=1}^{U_t} v_{g_{t,i}} \hat{\mu}_{f,p_{t,i}}(t). \tag{4.10}$$

If the set of files defined by (4.10) is not unique, ties are again broken arbitrarily.

After caching the selected files, the algorithm broadcasts the information about cache content $\mathcal{C}_t$. Then, the algorithm observes the users' requests for these files until the end of the time slot. Upon completion of the time slot, it updates the estimated demands and the counters of cached files.

4.6 Properties of Proposed Algorithm

4.6.1 Upper Bound on Regret

In this section, the performance of CAC is analyzed by determining its regret with respect to the oracle solution, as defined in (4.7). Specifically, the theorem presented below shows that the regret of CAC is sublinear in the time horizon T, i.e., it is shown that there exists $\gamma < 1$ for which $R(T) = O(T^{\gamma})$ holds. This bound on the regret guarantees that CAC converges to the oracle solution for $T \to \infty$, since then $\lim_{T\to\infty} \frac{R(T)}{T} = 0$

holds. The regret bound bases upon the assumption that the expected demand for a file is similar in similar contexts, meaning that users with similar contexts are likely to consume similar content. This assumption is natural since when the users' content preferences differ based on the users' contexts, one may divide the user population into segments of users with similar context and similar preferences. The similarity assumption is captured by the following Hölder continuity assumption, cf. Section 2.3.3.4.

Assumption 4.1 (Hölder continuity assumption)**.** *There exist* $L > 0$ *and* $0 < \alpha \leq 1$ *such that*

$$|\mu_f(\mathbf{x}) - \mu_f(\tilde{\mathbf{x}})| \leq L||\mathbf{x} - \tilde{\mathbf{x}}||_D^{\alpha} \tag{4.11}$$

holds for all $f \in \mathcal{F}$ *and for all* $\mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X}$*, where* $||\cdot||_D$ *denotes the Euclidean norm in* $\mathbb{R}^D$*.*

While Assumption 4.1 is needed for the analysis of the regret, it is important to note that CAC may also be applied to data which does not satisfy this assumption. However, a regret bound may not be guaranteed in this case.

The following theorem shows that the regret of CAC is sublinear in the time horizon T.

Theorem 4.1 (Bound for $R(T)$)**.** *Let* $K(t) = t^{\frac{2\alpha}{3\alpha+D}} \log(t)$*,* $t = 1, ..., T$*, and* $h_T = \lceil T^{\frac{1}{3\alpha+D}} \rceil$*. If CAC is run with these parameters and Assumption 4.1 holds true, the regret* $R(T)$ *is bounded by*

$$R(T) \leq mU_{\max}v_{\max}w_{\max}\Bigg(R_{\max}2^D|\mathcal{F}| \cdot (\log(T)T^{\frac{2\alpha+D}{3\alpha+D}} + T^{\frac{D}{3\alpha+D}}) + \frac{2R_{\max}}{(2\alpha+D)/(3\alpha+D)}T^{\frac{2\alpha+D}{3\alpha+D}} + 2LD^{\frac{\alpha}{2}}T^{\frac{2\alpha+D}{3\alpha+D}} + U_{\max}R_{\max}|\mathcal{F}|\frac{\pi^2}{3}\Bigg). \tag{4.12}$$

The leading order of the regret is hence $O\left(\log(T)T^{\frac{2\alpha+D}{3\alpha+D}}\right)$*.*

The proof can be found in Appendix A.5. The idea of the proof is as follows. First, the regret is decomposed into two terms, one term representing the regret due to exploration phases and one term representing the regret due to exploitation phases. Each of the two terms is then bounded separately. Bounding the first term works as follows. The loss due to selecting suboptimal files in exploration phases may be upper-bounded by a constant. Moreover, it is shown that the number of exploration phases is limited and can be bounded sublinearly in T, given an appropriate choice of the input parameters. Overall, this leads to a sublinear upper bound on the regret due

to exploration phases. The idea for bounding the second term is as follows. First, one distinguishes between two different types of exploitation phases, depending on whether the estimated demand $\hat{\mu}_{f,p_{t,i}}(t)$ of each file $f \in \mathcal{F}$ in each current hypercube $p_{t,i}$, $i = 1, ..., U_t$, is "close" to its expected value $\mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]$. Then, for exploitation phases in which the latter holds true, one can show that even if a suboptimal set of files is selected, the loss cannot be very large, but can in fact be bounded sublinearly in T, given an appropriate choice of input parameters. For the second type of exploitation phases, the loss due to selecting suboptimal files is upper-bounded by a constant. Moreover, one can show that the number of this type of exploitation phases is limited by a sublinear bound in T given an appropriate choice of the input parameters. Overall, this leads to a sublinear upper bound on the regret due to exploitation phases. Then, the overall regret bound follows by setting the appropriate input parameters.

Note that the proof technique used in this thesis is inspired by [KTvK18] and it is different from the one used in our original publications on context-aware proactive caching [MAvK16, MAvK17]. Using the new proof technique, the regret bound presented in Theorem 4.1 improves the regret bounds derived in [MAvK16, MAvK17] in some of the constant factors (while the order of the regret remains the same). The regret bound given in Theorem 4.1 is sublinear in the time horizon T. Hence, when T goes to infinity, CAC converges to the optimal cache content placement strategy that maximizes the expected number of weighted cache hits. Moreover, since Theorem 4.1 is applicable for any finite time horizon T, it characterizes CAC's speed of convergence. Finally, Theorem 4.1 shows that the regret bound for the general case with service differentiation is a constant multiple of the regret bound in the special case without service differentiation, in which $v_{\max} = 1$ and $w_{\max} = 1$ holds. Therefore, also in the case without service differentiation, the order of the regret is of order $O\left(T^{\frac{2\alpha+D}{3\alpha+D}} \log(T)\right)$.

4.6.2 Computational Complexity

Here, we analyze the computational complexity of the proposed algorithm within one time slot t as a function of the dimension D of the context space and of the library size $|\mathcal{F}|$. For this purpose, we identify the most computationally expensive procedures in the algorithm.

The complexity of line 6 in Algorithm 4.1 does neither grow with D nor $|\mathcal{F}|$, and hence the computational complexity of this line is $O(1)$. The observation of the user contexts grows as $O(D)$ since user contexts are vectors of length D, and each entry needs to be considered once. In Line 8, finding the hypercube in the partition of the context space

to which a user context vector belongs, has a complexity that grows as $O(D)$. This is because, one can give a separate index to each hypercube in each context dimension and then compute the index of the hypercube to which a context vector belongs in each of the D context dimension once and independently of the other dimensions. Computing the set of under-explored files in Line 9 has a complexity that grows as $O(|\mathcal{F}|)$ since each file has to be considered once. Lines 10 and 11 have a computational complexity that grows at most as $O(|\mathcal{F}|)$ by considering each file at most once. Line 12 neither grows with D nor $|\mathcal{F}|$, and hence its computational complexity is $O(1)$. Selecting m files in lines 13, 15-16 or 19 neither grows with D nor $|\mathcal{F}|$ and is hence of order $O(1)$. Ranking the files in lines 16 or line 19 has a complexity that grows as $O(|\mathcal{F}| \log |\mathcal{F}|)$ since at most $|\mathcal{F}|$ files need to be sorted [CLRS09]. Finally, the remaining lines neither grow with D nor $|\mathcal{F}|$, and hence their computational complexity is $O(1)$. Overall, the computational complexity of the proposed CAC algorithm grows hence as $O(D + |\mathcal{F}| \log |\mathcal{F}|)$, i.e., CAC has a computational complexity that grows linearly as a function of the number of context dimensions and log-linearly as a function of the file library size.

4.6.3 Memory Requirements

Here, we study the memory requirements needed for running the proposed context-aware proactive caching algorithm. The memory requirements of CAC mostly consist of the counters and estimates kept by the algorithm during its runtime, cf. [TvdS15a]. The algorithm keeps the counter $N_{f,p}$ and the estimate $\hat{\mu}_{f,p}$ for each pair consisting of a set $p \in \mathcal{P}_T$ and a file $f \in \mathcal{F}$. The number of files in $\mathcal{F}$ is $|\mathcal{F}|$. Moreover, if the input parameters from Theorem 4.1 are used, the number of sets in $\mathcal{P}_T$ is upper-bounded by $(h_T)^D = \lceil T^{\frac{1}{3\alpha+D}} \rceil^D \leq (1 + T^{\frac{1}{3\alpha+D}})^D$. Therefore, the required memory is upper-bounded by $2|\mathcal{F}|(1 + T^{\frac{1}{3\alpha+D}})^D$. This shows that the required memory is sublinear in the time horizon T. This means that when T goes to infinity, the algorithm approaches infinite memory requirements. However, since in practice, only the counter and estimate of those sets $p \in \mathcal{P}_T$ have to be kept to which at least one of the already arrived user contexts belonged, the actual number of kept counters and estimates may be much smaller than given by the upper bound.

4.6.4 Communication Requirements

Here, we study the communication requirements of CAC in one time slot t. In time slot t, the caching entity observes the contexts of the U_t currently connected users. If all context information is collected from the mobile devices (instead of from external

sources, see Section 4.3.3), the mobile device of each user needs to submit a vector of length D to the caching entity. Moreover, in time slot t, the caching entity informs the users about the current cache content by sending a broadcast, consisting of identifiers of the m currently cached files. This broadcast would also have to be sent using a proactive caching algorithm for caching at the edge which does not exploit context information [BG14b, BG14c, BG14a]. Hence, the only additional data transmission in CAC resulting from context awareness is the transmission of the U_t context vectors, and the amount of data that each mobile device needs to transmit is only linear in the dimension D of the context space. Moreover, even less data needs to be transmitted by the mobile devices if a part or all of the context information is collected from external sources (e.g., social media platforms).

4.7 Extensions

4.7.1 Multicast Transmissions

Multicasting has been shown to be beneficial in combination with caching [MAN14, PIST16]. CAC may be extended to include multicast transmissions in the following way. Instead of serving each request for a cached file by a unicast transmission, CAC may wait for several incoming requests and then serve requests for the same file by a multicast transmission. In detail, CAC could be extended as follows. Each time slot t is divided into a fixed number of intervals. During each of the intervals, the incoming requests for cached files are first monitored and accumulated. At the end of an interval, requests for the same file are served by a multicast transmission. Since there may be files which are only requested once within an interval and since knowledge about such low content popularity may already be available based on what has been previously learned, a request for a file with low estimated demand could, however, still be immediately served by a unicast transmission. In this way, unnecessary delays for the users are prevented in cases in which another request is unlikely and thus a multicast transmission may not be needed anyway. Finally, service differentiation could be taken into account when incorporating multicast transmissions into CAC. For example, CAC may always serve high-priority users via unicast transmissions such that their delay is not increased due to waiting times for multicast transmissions.

4.7.2 User Ratings

A content provider operating an infostation may want to cache content which is not only requested often, but which also receives high ratings from the users. Instead of selecting the cache content with respect to the expected demands $d_f(\mathbf{x})$ in order to maximize the number of (weighted) cache hits, CAC may be adapted to additionally take user ratings into account for cache content placement in the following way. Assume that a user may rate a content after she/he has requested and consumed the content. Further assume that the rating lies in a range $[r_{\min}, r_{\max}] \subset \mathbb{R}_+$. For a context $\mathbf{x}$, let $r_f(\mathbf{x})$ be the random variable describing the rating of a user with context $\mathbf{x}$ in case she/he requests file f and makes a rating thereafter. Then, we define the random variable

$$\tilde{d}_f(\mathbf{x}) := r_f(\mathbf{x}) d_f(\mathbf{x}), \tag{4.13}$$

which reflects both the demand and the rating of a user with context $\mathbf{x}$ for file f. By carefully designing the range of ratings, the content provider forms the trade-off between ratings and cache hits. When applying CAC with respect to $\tilde{d}_f(\mathbf{x})$ instead of $d_f(\mathbf{x})$, the algorithm additionally needs to observe the user ratings in order to learn content popularity in terms of ratings. If user ratings are always available, Theorem 4.1 applies and provides a regret bound of $O\left(T^{\frac{2\alpha+D}{3\alpha+D}} \log(T)\right)$.

However, users may not always submit a rating after consuming a content. When a user's rating for a file is missing, we assume that CAC does not update the counter and estimate of the corresponding file based on this user's content request. However, as a result, CAC may require a higher number of exploration phases. Hence, the regret of CAC is influenced by the willingness of the user population to submit ratings of requested content. Let $\beta \in (0,1)$ be the probability that a user submits a rating after requesting a file. Then, the regret of CAC is bounded as follows.

Theorem 4.2 (Bound for $R(T)$ for rating-based caching with missing ratings)**.** *Let $K(t) = t^{\frac{2\alpha}{3\alpha+D}} \log(t)$, $t = 1, ..., T$, and $h_T = \lceil T^{\frac{1}{3\alpha+D}} \rceil$. If CAC is run with these parameters with respect to $\tilde{d}_f(\mathbf{x})$, Assumption 4.1 from Section 4.6.1 holds true for $\mathbb{E}[\tilde{d}_f(\mathbf{x})]$ and a user submits a rating with probability β after requesting a file, the regret $R(T)$ is bounded by*

$$R(T) \leq m U_{\max} v_{\max} w_{\max} \Bigg(\frac{1}{\beta} R_{\max} 2^D |\mathcal{F}| \cdot (\log(T) T^{\frac{2\alpha+D}{3\alpha+D}} + T^{\frac{D}{3\alpha+D}}) + \frac{2R_{\max}}{(2\alpha+D)/(3\alpha+D)} T^{\frac{2\alpha+D}{3\alpha+D}} + 2LD^{\frac{\alpha}{2}} T^{\frac{2\alpha+D}{3\alpha+D}} + U_{\max} R_{\max} |\mathcal{F}| \frac{\pi^2}{3} \Bigg). \tag{4.14}$$

The leading order of the regret is hence $O\left(\frac{1}{\beta} T^{\frac{2\alpha+D}{3\alpha+D}} \log(T)\right)$.

The proof can be found in Appendix A.6. The proof of Theorem 4.2 works analogously to the proof of Theorem 4.1 applied to $\tilde{d}_f(\mathbf{x})$ instead of $d_f(\mathbf{x})$, by first dividing the regret into two terms representing the regret due to exploration and exploitation phases, and then bounding each of the terms separately. The regret due to exploitation phases then remains exactly the same. However, the regret due to exploration phases changes, since in case of rating-based caching with missing ratings, CAC does not update the counter and estimate when no rating is given for a requested file. Hence, the required number of exploration phases may increase. It can be shown that the expected number of exploration phases can be upper-bounded by $\frac{1}{\beta}$ times the number of exploration phases of the original case considered in Theorem 4.1. The rest of the proof works analogously to the proof of Theorem 4.1.

Note that the proof technique used in this thesis is inspired by [KTvK18] and it is different from the one used in our original publication on context-aware proactive caching [MAvK17]. Using the new proof technique, the regret bound presented in Theorem 4.2 improves the regret bound derived in [MAvK17] in some of the constant factors (while the order of the regret remains the same). Comparing Theorem 4.2 with Theorem 4.1, in case of rating-based caching with missing ratings, the regret of CAC is scaled up by a factor $\frac{1}{\beta} > 1$ in one summand. This factor corresponds to the expected number of content requests received by a user until the user submits one rating. Note that, however, the time order of the regret remains the same. Hence, CAC is robust under missing ratings in the sense that if a few users refuse to rate requested content, CAC still converges to the optimal cache content placement strategy when T goes to infinity.

4.7.3 Asynchronous User Arrival

Our model in Section 4.3.2 assumed that the set of connected users may change from one time slot to another, but that it remains static within each time slot. This means, that only those users may request files within a time slot that were connected to the caching entity in the beginning of that time slot. However, it is possible to extend CAC such that it takes into account that users connect to the caching entity asynchronously, i.e., within the time slots of CAC. If, after the context monitoring in the beginning of a time slot, a user immediately disconnects from the caching entity without requesting any file, that user should be excluded from learning. Hence, in CAC, the counters and estimates are not updated for immediately disconnecting users. If, after cache content placement within a time slot, a user connects to the caching entity, her/his content requests may be used for learning, even though her/his context has not been

considered in the caching decision of that time slot. Therefore, in CAC, the counters and estimates are updated based on this user's content requests.

4.7.4 Multiple Wireless Local Caching Entities

Practical caching systems contain multiple caching entities and each of the caching entities needs to learn its local content popularity. In such a network consisting of multiple caching entities, CAC, which was designed for cache content placement in a single caching entity, could be applied separately and independently by each caching entity. However, CAC may also be extended to the case that coverage areas of caching entities overlap. We call this extension the *context-aware proactive caching with area overlap* (CACao) algorithm. The main idea of CACao is that caching entities may learn content popularity faster by not only learning from their own cache hits, but also by learning from cache hits occurring at neighboring caching entities with overlapping coverage area. For this purpose, caching entities overhear cache hits from users in the intersection to neighboring coverage areas.

Specifically, CACao extends CAC as follows: CACao monitors user context and selects cache content in the same way as CAC does. However, using CACao, a caching entity not only observes its own cache hits (line 22 in Algorithm 4.1), but it overhears cache hits occurring at neighboring caching entities from users in the intersection to neighboring coverage areas. Subsequently, using CACao, the caching entity not only updates the counters and estimates of its own currently cached files (lines 23-27 in Algorithm 4.1), but it additionally updates the counters and estimates of files of which it overheard cache hits at neighboring caches. In this way, the caching entity may learn faster.

4.8 Numerical Results

4.8.1 Simulation Setup

We evaluate CAC by comparing its performance to several reference algorithms in simulations based on a real world data set from MovieLens [HK15]. MovieLens is an online movie recommendation system operated by the research group GroupLens at the University of Minnesota. The MovieLens 1M DataSet [Gro03] contains 1 000 209 ratings of 3952 movies made by 6040 MovieLens users within the years 2000 to 2003. Each

entry of the data set consists of an anonymous user ID, a movie ID, a rating (integer between 1 and 5) and a time stamp. Moreover, demographic information is available about the users' gender, age (in 7 categories), occupation (in 20 categories) and Zip-code.

In our simulations, we assume that the movie rating process from the MovieLens data set corresponds to a content request process of users connected to a wireless local caching entity (see [LXvdSL16b, LXvdSL16a] for a similar approach). Hence, when a user rates a movie at a certain point in time in the MovieLens data set, in our simulations, at the same point in time, this specific user requests this specific movie from either the caching entity (in case the movie is locally cached in the caching entity) or from the macro cellular network (in case the movie is not locally cached in the caching entity). Since users typically rate movies after watching them, this approach of translating a movie rating process to a content request process is reasonable. Moreover, for our simulations, we only use the data collected within the first year of the MovieLens data set, since around 94% of the ratings were provided within this time frame. Then, assuming that the caching entity updates its cache content on an hourly basis, we divide a year's time into 8760 time slots of one hour each (thereby setting $T = 8760$). Finally, we assign the content requests and corresponding user contexts to the 8760 time slots according to their time stamps, where we interpret each request as if it was coming from a separate user. Figure 4.4 depicts the content request process resulting from the described approach based on the MovieLens data set. Clearly, the content request process is bursty and flattens out towards the end.

Regarding the user context, we assume that at the beginning of a time slot, the caching entity has access to the context of all users responsible for the requests in the coming time slot. The context dimensions used in our simulations are gender and age. Note that we do not use occupation as context dimension in our simulations since by mapping occupations to a $[0, 1]$ variable, we would have to classify which occupations are more and which are less similar to each other.

In our simulations, each algorithm is run over the sequence of time slots $t = 1, ..., T$. All simulation results are obtained by averaging over 100 runs of the algorithms. Concerning the choice of input parameters, in ϵ-Greedy, we set $\epsilon = 0.09$ which is the value at which heuristically the algorithm on average performed best. Moreover, in CAC, we set the control function to $K(t) = \lambda_{\text{CAC}} \cdot t^{\frac{2\alpha}{3\alpha+D}} \log(t)$ with $\lambda_{\text{CAC}} = 1/(|\mathcal{F}|D)$. Compared to the control function in Theorem 4.1, the additional factor reduces the number of exploration phases which allows for better performance.

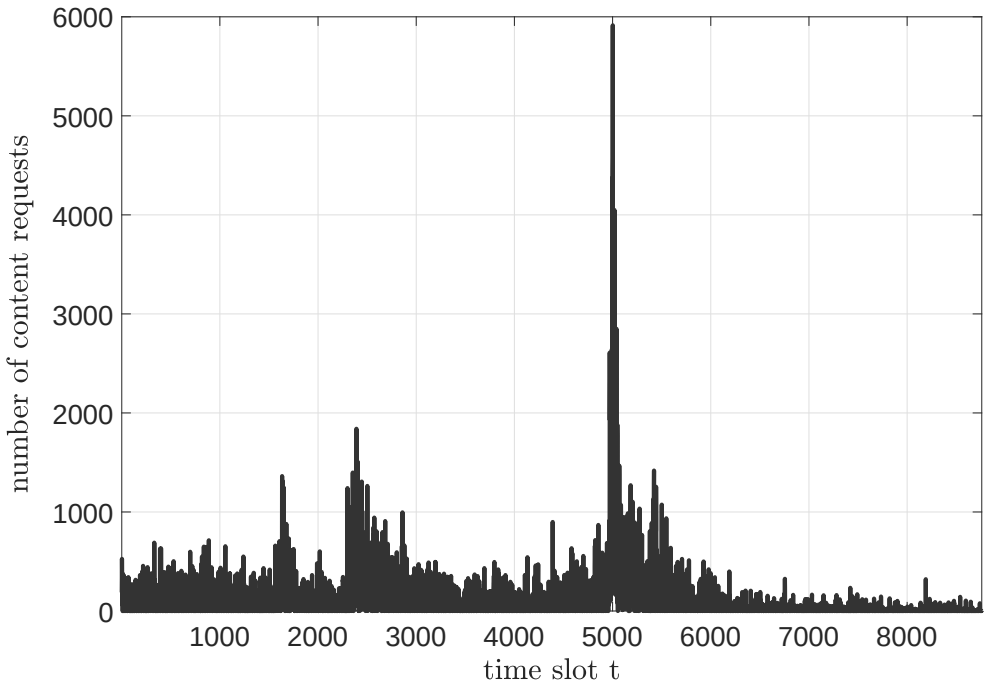


Figure 4.4. Number of content requests in used data set as a function of time slots. Time slots at an hourly basis.

4.8.2 Reference Algorithms

We evaluate CAC by comparing it with the following five reference algorithms.

- The *Oracle* has perfect a priori knowledge about the expected demand of each file in each context. In each time slot, the Oracle selects the top-m files as derived in (4.5).
- The *UCB* algorithm is based on an *upper confidence bound* (UCB) and consists of a variant of the UCB1 algorithm. UCB1 is a classical learning algorithm for the stochastic MAB problem with logarithmic regret order [ACBF02], cf. Section 2.3.3.3. However, it does not take context information into account, i.e., the logarithmic regret is with respect to the average expected demand over the whole context space. While the classical UCB1 takes one action per time slot, our modified UCB takes m actions per time slot, which corresponds to selecting m files.
- The ϵ-*Greedy* algorithm is a simple algorithm for the stochastic MAB problem, cf. Section 2.3.3.3, which learns from the history of reward observations [ACBF02],

but without taking into account context. While the ϵ-Greedy usually takes one action per time slot, we modified ϵ-Greedy to take m actions per time slot, which corresponds to selecting m files. In detail, in the considered caching scenario, the ϵ-Greedy caches a random set of m files with probability $\epsilon \in (0, 1)$ and the algorithm caches the m files with highest to m-th highest estimated demands with probability $(1 - \epsilon)$. The estimated demands are calculated as sample means of previous demands for cached files.

- *Myopic* is an algorithm taken from [BG14b], which is investigated since it is comparable to the well-known caching algorithm LRU, cf. Section 4.2. Myopic only learns from one time slot in the past. Starting with a random set of files, in each of the following time slots, Myopic discards the files which have not been requested in the previous time slot. The discarded files are then randomly replaced by other files.

- The *Random* algorithm caches a random set of files in each time slot.

4.8.3 Evaluation Metrics

The following metrics are used for evaluation purposes.

- The absolute performance of the algorithms is assessed based on the evolution of the *number of cache hits per time slot* and the *cumulative number of cache hits*.

- A relative performance measure is given by the *cache efficiency*, which describes the percentage of requests which can be served by cached files. Formally, the cache efficiency is defined as the ratio of cache hits compared to the overall demand, i.e.,

$$\text{cache efficiency in } \% = \frac{\text{cache hits}}{\text{cache hits} + \text{cache misses}} \cdot 100. \tag{4.15}$$

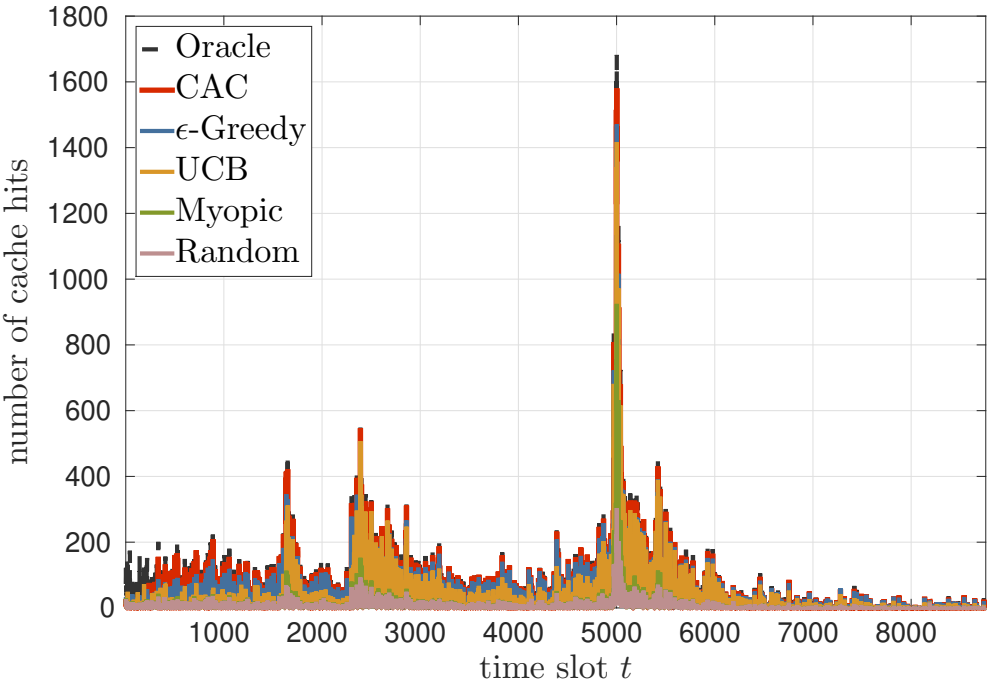
4.8.4 Results

First, we consider the case without service differentiation and we investigate the long-term behavior of CAC based on the following scenario. We assume that the caching entity may store up to $m = 200$ movies out of the $|\mathcal{F}| = 3952$ available movies so that the cache size corresponds to about 5% of the file library size [BG14b]. We run all

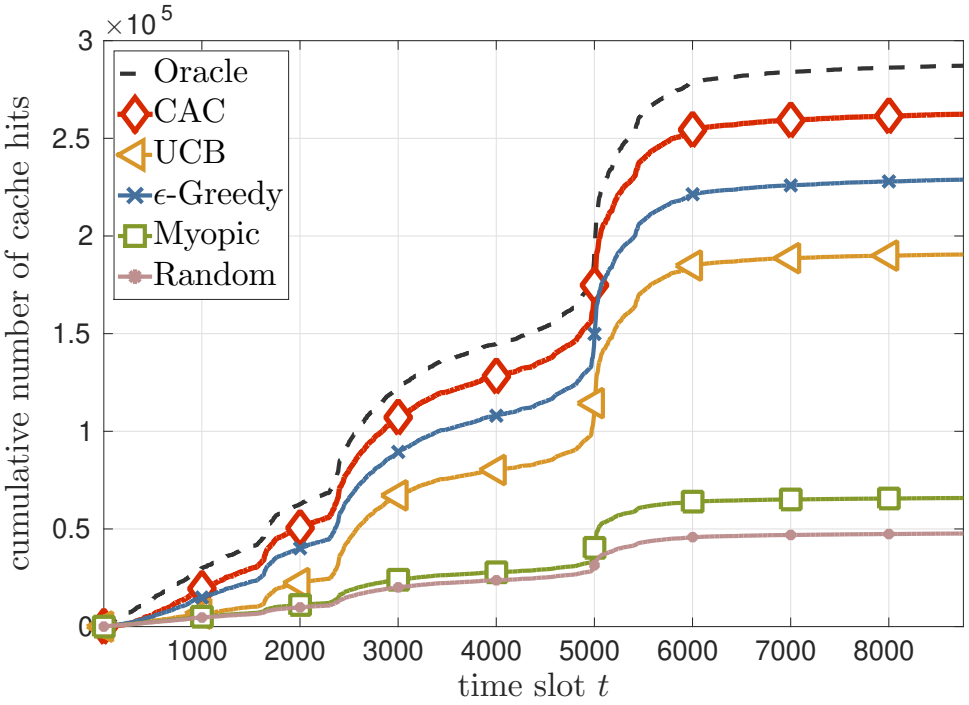
algorithms 100 times on the data set and average the results. Then, we study their performance as a function of time, i.e., over the time slots $t = 1, ..., T$. Figure 4.5(a) shows the number of cache hits per time slot achieved by the different algorithms as a function of time. As can be seen in Figure 4.5(a), the number of cache hits per time slot achieved by the different algorithms is bursty over time. This is due to the fact that the content request process itself is bursty (cf. Figure 4.4). Figure 4.5(a) shows that all algorithms roughly follow this bursty content request process. Since it is difficult to compare the results achieved by the different algorithms based on Figure 4.5(a), we plot the same data again, but this time based on the cumulative numbers of cache hits.

Figure 4.5(b) shows the cumulative number of cache hits up to time slot t achieved by the different algorithms as a function of time. As shown in Figure 4.5(b), the cumulative numbers of cache hits achieved by the different algorithms all follow the same trend, but on different levels. Random and Myopic achieve much smaller cumulative numbers of cache hits than the remaining algorithms. This is because Random does not learn at all and Myopic only learns from one time slot in the past. ϵ-Greedy and UCB show a better performance than Random and Myopic since they learn from the whole history of observed demands. Interestingly, it can be observed that ϵ-Greedy outperforms UCB, even though it uses a simpler learning strategy. The proposed algorithm CAC outperforms Random, Myopic, ϵ-Greedy and UCB since it not only learns from the whole history of observed demands, but additionally learns from context information. At the time horizon, the cumulative number of cache hits achieved by CAC corresponds to 1.14, 1.37, 3.98 and 5.50 times the cumulative numbers of cache hits achieved by ϵ-Greedy, UCB, Myopic and Random, respectively. Moreover, CAC yields a result close to the Oracle, which gives an upper bound to the other algorithms. In detail, the cumulative number of cache hits achieved by CAC corresponds to 0.91 times the cumulative number of cache hits achieved by the Oracle.

Next, we investigate the impact of the cache size m by varying it between 50 and 400 files. This corresponds to between about 1% and 10% of the file library size. The remaining parameters are kept as before and the results are again averaged over 100 runs of the algorithms. Figure 4.6 shows the overall cache efficiency achieved at the time horizon T, i.e., the cumulative number of cache hits up to T is normalized by the cumulative number of requests up to T, as a function of the cache size m. The overall cache efficiency of all algorithms is increasing with increasing cache size. Moreover, the results indicate that Random and Myopic perform by far worse compared to the other algorithms. Again, ϵ-Greedy and UCB are outperformed by the proposed CAC. In detail, averaged over the considered range of cache sizes, the average cache efficiency of CAC is 28%, compared to average cache efficiencies of 25%, 21%, 8% and 6%



(a) Number of cache hits per time slot.



(b) Cumulative number of cache hits.

Figure 4.5. Time evolution of algorithms for $m = 200$.

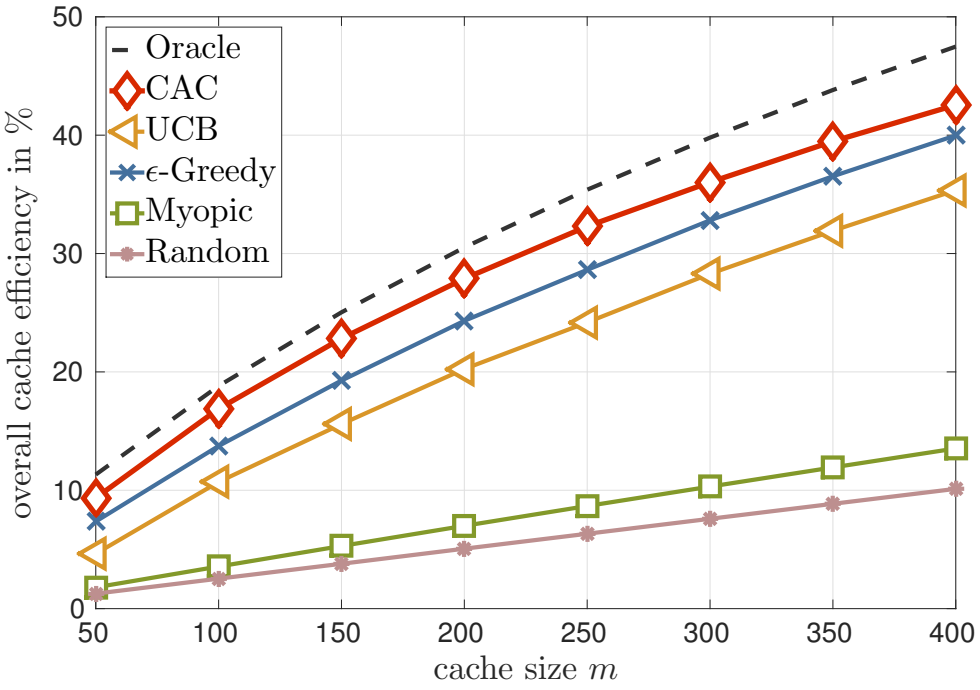


Figure 4.6. Overall cache efficiency at T as a function of cache size m.

achieved by ϵ-Greedy, UCB, Myopic and Random, respectively. Moreover, regarding the achieved cumulative numbers of cache hits, based on which the cache efficiencies in Figure 4.6 were computed, CAC achieves up to 27% more cache hits than the next best algorithm ϵ-Greedy in the considered range of cache sizes. Finally, with its average cache efficiency of 28%, CAC lies not far away from the average cache efficiency of 32% achieved by the Oracle.

Now, we consider a case of service differentiation, in which two different service groups 1 and 2 with weights $v_1 = 5$ and $v_2 = 1$ exist. Since service group 1 represents a higher value, users of this service group should be prioritized in caching decisions. In our simulations, we randomly assign 20% of the users to service group 1 and classify all remaining users as service group 2. Then, we adjust each algorithm to take service differentiation into account by incorporating the weights corresponding to the service groups into the algorithms. The results are again averaged over 100 runs of the algorithms. Figure 4.7 shows the cumulative number of weighted cache hits up to time slot t for a cache size of $m = 200$ as a function of time. A comparison with Figure 4.5(b) shows that the general behavior is similar to the case without service differentiation. The cumulative numbers of weighted cache hits achieved by the different algorithms all follow the same trend, but on different levels. Again, Random and Myopic achieve much lower performance compared to the remaining algorithms.

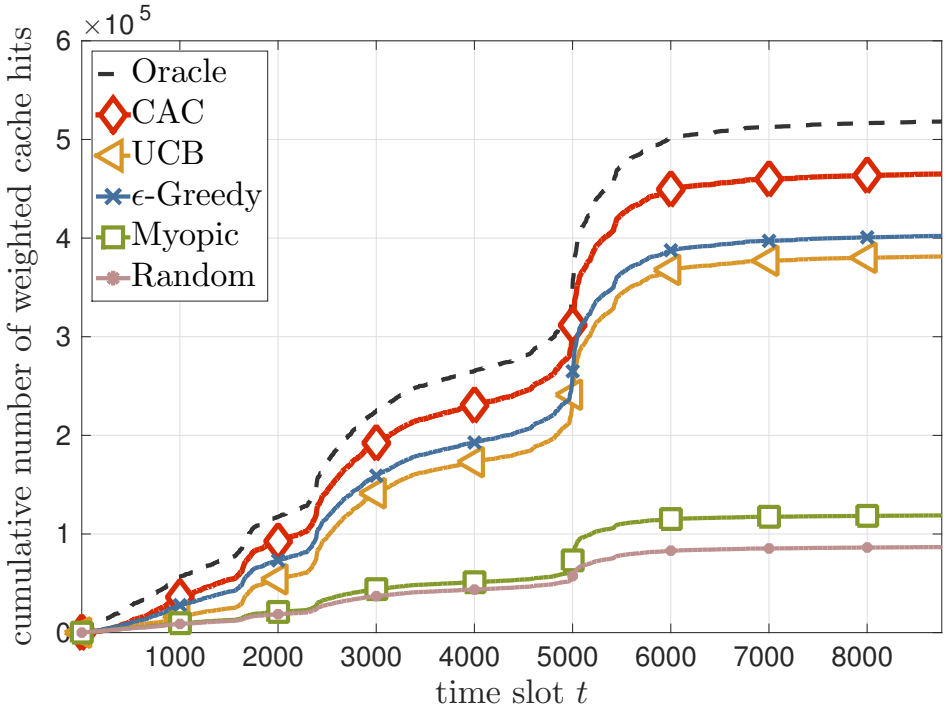


Figure 4.7. Cumulative number of weighted cache hits for $m = 200$ as a function of time.

ϵ-Greedy and UCB show a better performance than Random and Myopic. However, all of them are outperformed by the proposed CAC. At the time horizon, the cumulative number of weighted cache hits achieved by CAC corresponds to 1.15, 1.21, 3.91 and 5.36 times the cumulative numbers of weighted cache hits achieved by ϵ-Greedy, UCB, Myopic and Random, respectively. In addition, CAC achieves results still close to the upper bound provided by the Oracle. The cumulative number of weighted cache hits achieved by CAC corresponds to 0.89 times the cumulative number of cache hits achieved by the Oracle.

Finally, we investigate a scenario of multiple caching entities and compare the performance of the two proposed algorithms CAC and CACao. We consider two caching entities and divide the used data set as follows. We assume that a fraction $o \in [0, 0.3]$ of randomly selected requests is made in the intersection of the two coverage areas, whereas the remaining requests are randomly assigned to either the one or the other caching entity since they are considered to be made by users solely connected to one caching entity. The parameter o hence can be seen as a measure of the overlap between the caching entities. Now, on the one hand, we run CAC separately on each caching entity and, on the other hand, we run CACao on both caching entities. Figure 4.8 shows the cumulative number of cache hits achieved in sum by the two caching entities at the

time horizon T as a function of the overlap parameter o. As expected, CAC and CACao perform identically for non-overlapping coverage areas (i.e., when $o = 0$). Moreover, the numbers of cache hits achieved by both CAC and CACao increase with increasing overlap. This is because it is more likely that users in the intersection of the coverage areas can be served since these users have access to both caches. Hence, even though the caching entities do not coordinate their decisions on cache content, more cache hits occur. Comparing CACao with CAC, it can be seen that for up to 25% of overlap ($o \leq 0.25$), CACao outperforms CAC. The reason is that by overhearing cache hits at the neighboring caching entity, both caching entities learn content popularity faster using CACao. However, CAC yields higher numbers of cache hits for very large overlap ($o > 0.25$). This is because when applying CACao in case of a very large overlap, neighboring caching entities overhear such a large number of cache hits, that they learn very similar content popularity distributions. Therefore, it is likely that their caches contain the same files after some time. In contrast, a higher diversity in cache content is maintained over time when CAC is applied. In general, further gains in the number of cache hits could be achieved by jointly optimizing the cache content of all caching entities. However, this would either require coordination among the caching entities or a central decision agent selecting the cache content of all caching entities, which would result in a high communication overhead. In contrast, the heuristic approach used by CACao neither requires coordination nor communication between caching entities and yields good results for reasonably sized overlaps.

4.9 Conclusions

In this chapter, we have studied how to exploit caching resources in order to save communication resources in wireless networks. In detail, we have investigated the problem of context-aware proactive caching at the edge for maximizing the number of cache hits under missing knowledge about content popularity. We have proposed a model for context-aware proactive caching that allows different content to be favored by different users and that takes into account that the content popularity depends on the user's context. Moreover, we have taken a machine-learning-based approach by modeling the problem as a contextual MAB problem. We have proposed an online learning algorithm for context-aware proactive caching using a decentralized architecture of decision making. Using this algorithm, the controller of a local cache at the edge of the wireless network may learn context-specific content popularity online by regularly observing context information of connected users, updating the cache content and observing cache hits subsequently.

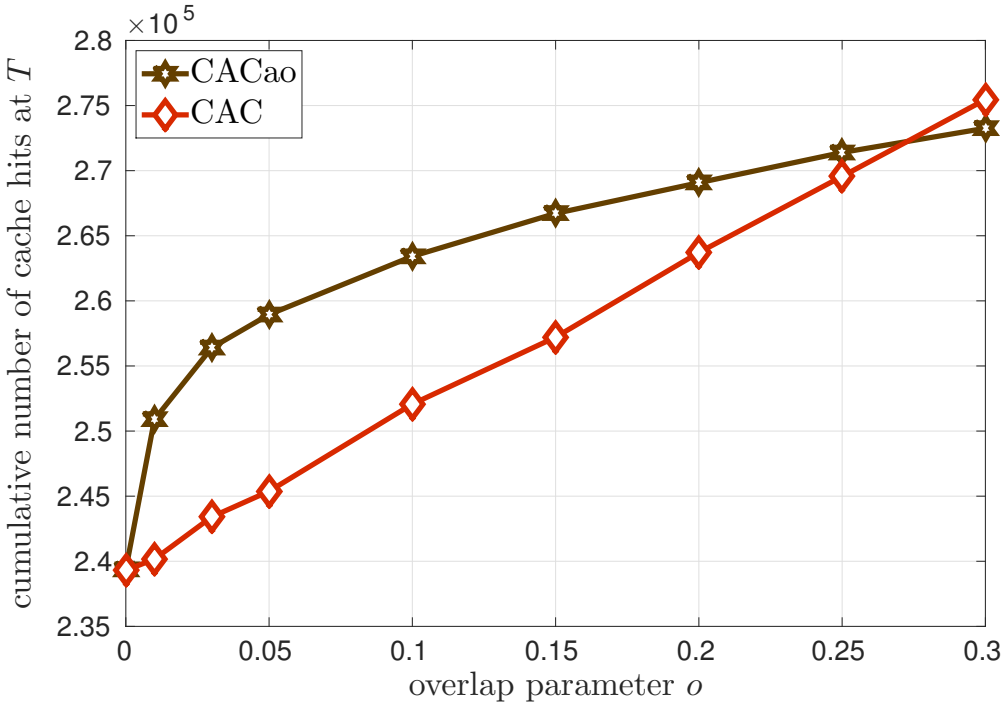


Figure 4.8. Cumulative number of cache hits at T as a function of the overlap parameter o.

The computational complexity of the proposed algorithm has been shown to grow linearly as a function of the number of context dimensions and log-linearly as a function of the file library size. Moreover, the memory and communication requirements of the proposed algorithm have been analyzed and extensions to practical requirements have been made. Moreover, we have derived a sublinear upper bound on the regret, which analytically bounds the loss of the proposed algorithm with respect to an oracle that executes the optimal cache content placement strategy given a priori knowledge on content popularity. The regret bound characterizes the learning speed and proves that the proposed algorithm converges to the optimal cache content placement strategy. Simulations based on real data have shown that, depending on the cache size, the proposed algorithm achieves up to 27% more cache hits than the best algorithm taken from the literature by exploiting contextual information for proactive cache content placement.

Chapter 5

Mobile Crowdsourcing

5.1 Introduction

In this chapter, we consider *mobile crowdsourcing* (MCS), a technique that exploits *user resources* for task completion. Specifically, task owners may outsource tasks via an intermediary *mobile crowdsourcing platform* (MCSP) to a set of mobile users, or *workers*, who are free to decide whether or not to complete assigned tasks [RZZS15]. Recently emerging MCS applications exploit that online mobile users may complete tasks anytime and anywhere on the go by considering *non-spatial* tasks, i.e., tasks that do not require the workers to be at a certain location for task completion. MCS is hence a technique that may enable different stakeholders to leverage human intelligence for task completion [RZZS15]. Clearly, different workers may have different interests and capabilities, and therefore not all of them may perform equally well on a given task [GS14]. In order to achieve the best possible outcome on a given task under a possibly limited budget by the task owner, the most suitable workers should hence be assigned to a task [TTSRJ14]. As described in Section 1.3.4, selecting the best workers for each task in an MCS application requires knowledge about the performance of each worker in terms of her/his acceptance rate and quality, but such knowledge is typically not available a priori and hence needs to be learned [SC17, ZC17, HV12, TTSRJ14, HZL16, uHC14]. Moreover, a worker's performance may depend not only on the specific task, but also on the worker's current context [GS14], and this dependency may be of non-linear nature. Finally, due to communication overhead and privacy concerns of workers, it may be required to protect personal worker context locally instead of sharing it with the central MCSP [TGFS17, GWG+16], which makes it even more difficult for the MCSP to select the most suitable workers.

Hence, we investigate the problem of maximizing the worker performance in an MCS application with non-spatial tasks, taking into account the following aspects:

(i) A priori, there is no knowledge available about each worker's individual performance in terms of her/his acceptance rate and quality.

(ii) Tasks of different types may arrive to the MCSP.

(iii) A worker's performance may depend in a possibly non-linear fashion on both the task and the current worker context.

(iv) A worker's personal context should be kept locally in order to keep the communication overhead small and to ensure the worker's privacy.

In this chapter, we propose a *machine-learning-based approach* and a *hierarchical architecture of decision making*. We take a machine-learning-based approach since the expected performance of a worker is not known in advance and hence needs to be learned. Moreover, we take a hierarchical architecture of decision making since the workers' personal contexts should be kept locally in order to keep the communication overhead small and to protect the workers' privacy. In detail, we propose a context-aware hierarchical online learning algorithm for worker selection in MCS applications based on a contextual MAB model. The algorithm is split into two parts. On the one hand, a *local controller* (LC) in each of the workers' mobile devices is responsible for learning its corresponding worker's context-specific performance in terms of acceptance rate and quality online over time, by regularly observing the worker's context, her/his decisions to accept or decline tasks and the quality in completing tasks. Based on these observations, the LC regularly estimates the worker's context-specific performance and sends this estimate to the MCSP. On the other hand, the MCSP is responsible for the worker selection, which it does based on performance estimates received from the LCs. This hierarchical coordination approach enables the LCs to learn context-specific worker performances and it enables the MCSP to select suitable workers without having access to the workers' personal contexts, which stay locally with the LCs. Moreover, workers receive personalized task requests based on their interests and skills, while at the same time the number of quality assessments, which are needed to observe worker performances, but may be costly, is kept low.

This chapter presents work originally published by the author in [KTvK18]. Compared to [KTvK18], in this thesis, the ideas of the mathematical proofs are additionally summarized and discussed within the main body of text, while the full mathematical proofs are given in the appendices. Furthermore, in this thesis, the computational complexity of the proposed algorithm is analyzed. The remainder of this chapter is organized as follows. Section 5.2 provides a detailed review of the state of the art on decision making for crowdsourcing systems. The system model for context-aware worker selection in an MCS application is introduced in Section 5.3. In Section 5.4, we give a formal problem formulation of context-aware worker selection for maximizing the worker performance in MCS under missing knowledge about expected worker performance and show that the problem can be understood as a contextual MAB problem. In Section 5.5, we

propose a context-aware hierarchical online learning algorithm for worker selection in MCS. Properties of the proposed algorithm are discussed in Section 5.6. Specifically, by deriving an analytical upper bound on the regret of the proposed algorithm, it is shown that the algorithm converges to the optimal worker selection strategy. In Section 5.7, the performance of the proposed algorithm is demonstrated numerically. Section 5.8 concludes this chapter.

5.2 State of the Art

In this section, we give a detailed review of the state of the art on decision making for *crowdsourcing* (CS) systems. We start by shortly discussing strategic behavior in CS and quality estimation in CS in case of missing ground truth. Then, we discuss in detail the related work on CS without strategic behavior and with available ground truth since the problem considered in this thesis is of this type.

One line of work on CS systems deals with strategic behavior of workers and task owners in CS systems. Such strategic behavoir could concern pricing and the effort spent in task completion [AvdS16]. Another line of work on CS deals with how to estimate the quality of a completed task in case of missing ground truth, for instance, using online learning for quality estimation [LL17].

In contrast, many related works consider task assignment and worker selection problems in CS systems without taking into account strategic behavior and under the assumption that it is possible to assess the quality of a completed task. Also the problem considered in this thesis belongs to this type. Therefore, in the remainder of this section, we review and discuss this type of related work on worker selection in CS. This review complements the short review presented in Section 1.3.4 by discussing in detail the works introduced in Table 1.3.

In this thesis, worker selection in MCS for non-spatial tasks is considered. Note that the related work discussed below not only covers decision making in MCS systems [RZZS15], but also in general web-based CS systems [DRH11] as well as in spatial CS systems [ZH16], since these works are also relevant for comparison with our approach. Also note that in general, task assignment and worker selection problems in CS systems are often modeled as problems of online decision making due to the dynamic nature of tasks and/or workers typically arriving over time [SV13].

We start by discussing related work on general CS systems. Ref. [HV12] proposes a competitive online task assignment algorithm in the *server assigned tasks* (SAT) mode

that learns the skills of sequentially appearing workers in order to maximize the utility of a task owner on a given set of task types, with finite number of tasks per type. In [TTSRJ14], an online task assignment algorithm in SAT mode with sublinear regret is proposed for expert CS. Based on a bounded MAB model, the algorithm aims at maximizing the utility of a budget-constrained task owner under uncertainty about the skills of a finite set of expert workers with known different prices and limited working time. The algorithm hence learns the average skill of a worker. Ref. [SC17] proposes a real-time algorithm in SAT mode for finding the top-k workers for sequentially arriving tasks. In a first step, tasks are categorized offline into different types and the similarity between a worker's profile and each task type is computed. Then, in real time, the top-k workers are selected for a task based on a matching score that takes the similarity scores and historic worker performance into account. The performance estimates are proposed to be updated offline in batches. Ref. [AVC11] proposes methods for learning a worker preference model that can be used for personalized *task recommendation* (TR) in the *worker selected tasks* (WST) mode. The proposed methods use the history of worker preferences on different tasks.

Among the related work on MCS systems, Ref. [GWG+16] proposes algorithms for optimal TR in WST mode that take into account trade-offs between the privacy of worker context, the utility to recommend the best tasks and the efficiency in terms of communication and computation overhead. In their approach, a server performs TR based on a generalized context shared by the worker. The statistics used for TR are collected offline via a proxy which ensures differential privacy guarantees. This approach allows to flexibly adjust the shared generalized context and makes TRs based on offline statistics and generalized, instead of individual, worker context. In [HZL16], an online learning algorithm for mobile crowdsensing in SAT mode is proposed to maximize the revenue of a budget-constrained task owner by learning the sensing values of workers with known prices.

A taxonomy for spatial CS was first introduced in [KS12], where a location-entropy based algorithm for SAT mode is proposed to maximize the number of task assignments under uncertainty about task and worker arrival processes. The server decides about task assignment based on centrally collected knowledge about the workers' current locations. In [TSK15], the above framework is extended to maximize the quality of assignments under varying worker skills for different task types. In both [KS12] and [TSK15], worker context is collected centrally. Moreover, it is assumed that workers always accept assigned tasks within certain known bounds and that worker skills are known a priori. In [uHC14], an online task assignment algorithm is proposed for spatial CS with SAT mode for maximizing the expected number of accepted tasks by selecting

appropriate workers for sequentially arriving tasks. The problem is modeled as a contextual MAB problem. Then, the LinUCB algorithm, cf. Section 2.3.3.4, is adapted to the problem by assuming that the acceptance rate of a worker is a linear function of the worker's distance to the task location and of the task type. However, such a linearity assumption is restrictive and it especially may not hold in MCS with non-spatial tasks. Ref. [TGFS17] proposes an algorithm for privacy-preserving spatial CS in SAT mode. Using differential privacy and geocasting, the algorithm preserves worker context in terms of their locations while optimizing the expected number of accepted tasks. The algorithm is based on the assumption that the workers' acceptance rates are identical and known. Ref. [ZC17] proposes exact solutions and approximation algorithms for acceptance maximization in spatial CS with SAT mode. The algorithms are performed offline for given sets of available workers and tasks based on a probability of interest for each pair of worker and task. The probabilities of interest are computed beforehand using maximum likelihood estimation.

The above discussed related work on decision making for CS systems can be categorized as follows. As seen above, different works consider different *types of CS*, such as general CS [HV12, TTSRJ14, SC17, AVC11], MCS [GWG+16, HZL16] or spatial CS [KS12, TSK15, uHC14, TGFS17, ZC17]. Moreover, the works differ regarding the considered *task assignment mode*, some works considering the SAT mode [HV12, TTSRJ14, SC17, HZL16, KS12, TSK15, uHC14, TGFS17, ZC17], others considering the WST mode [AVC11, GWG+16], where each mode has its specific advantages and disadvantages, as discussed in Section 1.3.4. Some related works assume that the *workers' performances are known in advance* (e.g., in terms of acceptance rates and quality) [KS12, TSK15, TGFS17], while others consider the more realistic case of task assignment *under missing knowledge about worker performance* [HV12, TTSRJ14, SC17, AVC11, GWG+16, HZL16, uHC14, ZC17]. In the latter case, different *types of machine-learning-based approaches* are applied, using, for instance, offline learning [GWG+16, ZC17], batch learning [SC17] or online learning [HV12, TTSRJ14, AVC11, HZL16, uHC14], the last approach being able to better adapt to varying worker performances. While some of the learning algorithms are only numerically evaluated [SC17, AVC11, GWG+16, ZC17], other related works additionally provide analytical *regret bounds* for their learning algorithms [HV12, TTSRJ14, HZL16, uHC14]. While some works consider only one type of tasks [TTSRJ14, HZL16, TGFS17], others take into account that *different types of tasks* may occur in MCS applications [HV12, SC17, AVC11, GWG+16, KS12, TSK15, uHC14, ZC17]. Moreover, while some works are unaware of worker context [HV12, TTSRJ14, SC17, AVC11, HZL16, ZC17], others take *worker context* into account [GWG+16, KS12, TSK15, uHC14, TGFS17], which is important since at least in

the MCS applications considered in this thesis, worker context may affect worker performance. However, even among the works which take worker context into account, only few consider such *context-specific worker performance* [GWG+16, uHC14, TGFS17]. Finally, among the works which take worker context into account, only some consider that *personal worker context needs to be protected* due to overhead and privacy reasons (i.e., keeping it completely locally, or sharing only generalized context information with the MCSP) [GWG+16, TGFS17].

Table 5.1 gives an overview of the discussed related work on worker selection in CS systems and provides a detailed comparison of the related work with the proposed algorithm. Compared to the related work, cf. Table 5.1, we propose a context-aware hierarchical online learning algorithm for worker selection in MCS for non-spatial tasks that for the first time *jointly* considers the following aspects:

(i) The proposed algorithm *does not assume a priori knowledge about worker performance*, but learns worker performance in terms of acceptance rate and quality *online* to maximize the average worker performance over time without requiring a training phase. Since the algorithm learns in an online fashion, it adapts and improves the worker selection over time and can hence achieve good results already during run time. By deriving an *upper bound on the regret* of the algorithm, we provide performance guarantees and prove that the algorithm converges to the optimal worker selection strategy.

(ii) The proposed algorithm allows *different task types* to occur. The concept of task context is used to describe the features of a task, such as its required skills or equipment.

(iii) The proposed algorithm allows that a worker's performance depends (in a possibly non-linear fashion) on both the task context and on the *worker context*, such as the worker's current location, activity, or device status. The proposed algorithm learns this *context-specific worker performance*.

(iv) The proposed algorithm is split into two parts, one part executed by the MCSP, the other part by LCs located in each of the workers' mobile devices. Based on this new hierarchical coordination approach between the MCS and LCs, the proposed approach combines the advantages of the SAT and the WST mode for task assignment. In particular, suitable workers can be selected for each task while at the same time, the workers' *personal contexts are protected* by keeping them locally, which keeps the communication overhead small and ensures the workers' privacy. Moreover, workers receive personalized task requests based on

Table 5.1. Related work on worker selection in CS and detailed comparison with proposed algorithm.

Reference	[HV12]	[TTSRJ14]	[SC17]	[AVC11]	[GWG+16]	[HZL16]	[KS12], [TSK15]	[uHC14]	[TGFS17]	[ZC17]	This work
CS type	General	General	General	General	Mobile	Mobile	Spatial	Spatial	Spatial	Spatial	Mobile, non-spatial
Task assignment mode	SAT	SAT	SAT	WST + TR	WST + TR	SAT	SAT	SAT	SAT	SAT	proposed
Worker performance	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Known	Unknown	Known	Unknown	Unknown
Type of learning	Online	Online	Batch	Online	Offline	Online	N/A	Online	N/A	Offline	Online
Regret bounds	Yes	Yes	No	No	No	Yes	N/A	Yes	N/A	No	Yes
Different task types	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes
Worker context-aware	No	No	No	No	Yes	No	Yes	Yes	Yes	No	Yes
Context-specific performance	No	No	No	No	Yes	No	No	Yes	Yes	No	Yes
Worker context protected	N/A	N/A	N/A	N/A	Yes	N/A	No	No	Yes	N/A	Yes

their interests and skills, while the number of possibly costly quality assessments is kept low.

5.3 System Model

5.3.1 Introduction

In this section, we propose a model for context-aware worker selection in an MCS application for non-spatial tasks. In accordance with Section 2.2.1, the proposed overall model consists of the following five components:

(i) The *network model* contains a model of an MCS application and introduces the MCSP that serves as intermediary between task owners and workers.

(ii) A *context model* is defined, which describes the side information about the worker and the task that may impact a worker's performance in terms of her/his acceptance rate and quality when completing a task.

(iii) As performance criterion to be maximized, the cumulative worker performance is considered. Since the worker performance depends on the worker and task context, a *model of context-specific worker performance* is formulated, which explicitly allows worker performance to be a (possibly non-linear) function of the task context and of the worker context.

(iv) A hierarchical *architecture of decision making* is proposed. Responsible for information collection and decision making are a set of LCs and the MCSP, respectively. Hence, a model of an LC is proposed.

(v) An *action model* is formulated, which determines the different choices of the MCSP, namely, which workers should be requested to complete a task.

5.3.2 Network Model

We consider an MCS application for non-spatial tasks. The MCS application is coordinated by an MCSP. By $\mathcal{W}$, we denote the set of $W := |\mathcal{W}|$ *workers*. A worker is a mobile user who has set up the MCS application in her/his mobile device. A worker can

be in the following two modes. A worker is called *available* if the MCS application on her/his device is running. In this case, the MCSP may request the worker to complete a task, which the worker may then accept or decline. A worker is called *unavailable* if the MCS application on her/his device is turned off.

Task owners may place non-spatial tasks of different types into the MCSP. We assume that tasks arrive at the MCSP sequentially and we denote the sequentially arriving tasks by $t = 1, ..., T$, where T denotes the total number of tasks. The submission of a task t is accompanied by a tuple $(b_t, \mathbf{c}_t)$, where $b_t > 0$ denotes the budget that the task owner is willing to pay for this task and $\mathbf{c}_t$ denotes the task context. The task context is a vector containing information about the task, and will be formally introduced in Section 5.3.3. The task owner is charged by the MCSP for each worker that completes the task after being requested by the MCSP. Specifically, we assume that the MCSP charges the task owner a fixed price $e_t \in [e_{\min}, e_{\max}]$ per worker that completes task t, where $e_{\min} > 0$ and $e_{\max} \geq e_{\min}$ correspond to lower and upper price limits, respectively. The price e_t may depend on the task context $\mathbf{c}_t$ and is determined based on a fixed and transparent context-specific price list held by the MCSP. We assume that for each task t, the budget b_t satisfies $b_t \in [e_t, We_t]$, i.e., the budget is sufficient to pay at least one and at most W workers for completing the task. Moreover, we assume that the sequence $\{b_t\}_{t=1,...,T}$ is not influenced by decisions about worker selection taken by the MCSP. Based on the budget b_t and the price e_t, the MCSP computes the maximum number $m_t := \lfloor \frac{b_t}{e_t} \rfloor \in \{1, ..., W\}$ of workers who should complete the task.

Following [HV12, TTSRJ14, HZL16], we assume that each task has the following properties:

- Depending on the budget and price of a task, the task owner would like to receive replies from possibly several workers who completed the task.
- It is possible to assess the quality of a single worker's reply.
- The qualities of different workers' replies are independent.
- The qualities of different workers' replies are additive, i.e., if two workers complete the task and their respective qualities are A and B, the task owner receives a total quality of $A + B$.

Such tasks belong to the class of *crowd solving* tasks [GS14], examples being translation and retrieval tasks [HV12]. Note that we assume that there exists a type of quality assessment which can be used to evaluate how well a worker performed in completing a

task. Among possible types of quality assessment are manual quality ratings from task owners, or an automatic quality assessment using either local software in the mobile devices or using the resources of a cloud. The proposed model and algorithm are agnostic to the specific type of used quality assessment. In general, however, quality assessments may be costly. Therefore, the number of worker performance observations may have to be kept low in order to keep the cost for quality assessment low.

At the arrival of a task, not all workers may be currently available since workers may arbitrarily turn on and off their MCS application in their mobile device over time. Therefore, the set of available workers may change dynamically over time. We denote the set of workers available at the arrival of task t by $\mathcal{W}_t \subseteq \mathcal{W}$, as defined by $\mathcal{W}_t := \{i : \text{ worker } i \text{ is available at arrival time of } t\}$, where $W_t := |\mathcal{W}_t| \in \{1, ..., W\}$ denotes the number of available workers at the arrival of task t. Our only assumptions on the nature of the arrival process of the availability of workers are that (i) for each arriving task, at least one worker is available and that (ii) the sequence $\{\mathcal{W}_t\}_{t=1,...,T}$ is not influenced by decisions about worker selection taken by the MCSP.

If sufficiently many workers are currently available at the arrival of task t, the MCSP requests m_t workers to complete the task. Note that each task is only processed once by the MCSP, even if not all m_t requested workers complete the task. Therefore, if not all of the m_t requested workers complete task t, the MCSP charges the task owner only for the actual number of workers that completed the task since only these workers are compensated. Moreover, it may also happen that fewer than m_t workers are currently available at the arrival of task t. In this case, the MCSP requests all available workers to complete the task. To sum up, the MCSP aims at selecting a subset of $\min\{m_t, W_t\}$ workers which maximizes the worker performance for the task. Figure 5.1 shows an illustration of the considered model, where a task arrives at an MCSP, which in turn needs to select an appropriate subset of currently available workers for the task.

5.3.3 Context Model

Appropriate worker selection requires knowledge about worker performance. Since a worker can have certain preferences regarding the types of tasks she/he likes, the worker may hence have different acceptance rates on different tasks. Moreover, since a worker has a certain set of skills, she/he may hence provide different quality when completing different tasks. Therefore, a worker's performance in terms of acceptance rate and quality may depend on the features of the task. The features of a task may be summarized under the term *context*. Possible features of a task could be the skills

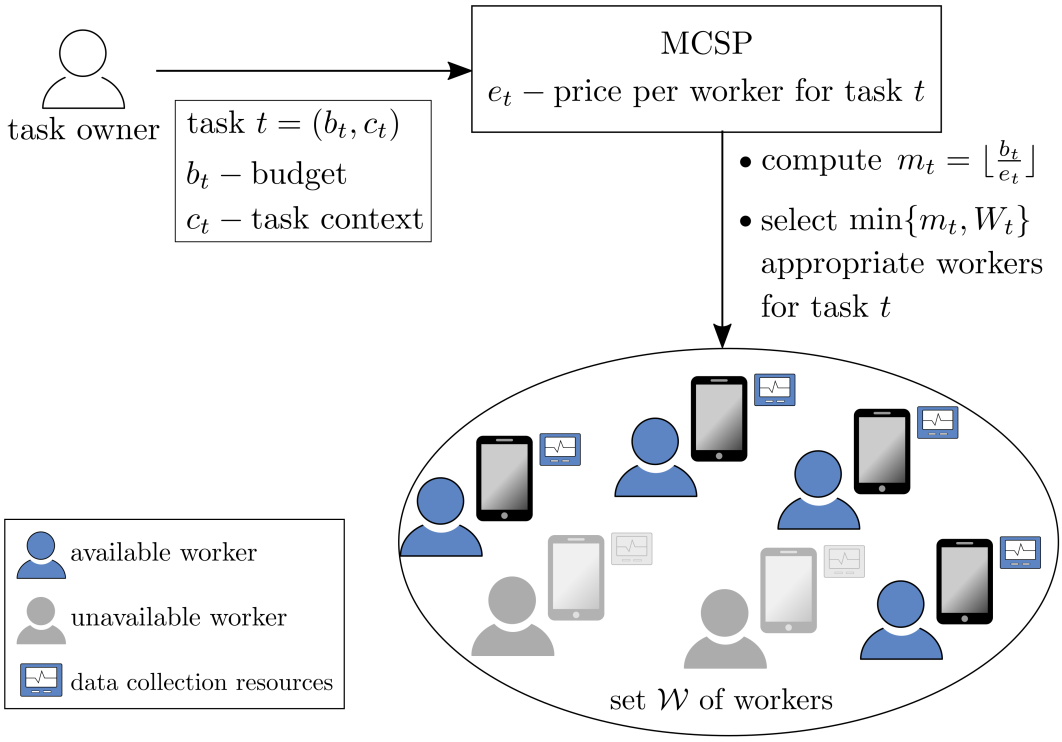


Figure 5.1. Network model.

needed (e.g., the required levels of creativity or analytical skills) or the equipment required (e.g., a camera or a specific application) to complete the task.

Formally, we model task context as follows. We denote the number of task context dimensions by C and we denote the C-dimensional task context space by $\mathcal{C}$. The context space $\mathcal{C}$ is assumed to be bounded and can hence be set to $\mathcal{C} := [0,1]^C$ without loss of generality. Hence, we assume that the feature information about a task is described in terms of C context dimensions. In each of the C context dimensions, a task is classified via a value between $[0,1]$, such that the overall task context is hence a vector in $[0,1]^C$. In practice, a pre-processing may be needed to bring the feature information into the correct format (e.g., the required levels of creativity or analytical skills may be translated to continuous values between 0 and 1; whether a camera or a specific application is needed may be encoded as binary 0 or 1).

The performance of a worker in terms of acceptance rate and quality may not only depend on the characteristics of the specific task to be completed, but also on the worker's personal current situation and environment. Possible relevant personal context dimensions could be the worker's current location (in terms of geographic coordinates) or the time of day [GS14], the type of location (e.g., at home, in a coffee shop), the worker's current activity (e.g., commuting, working) or the current device status (e.g.,

battery state, type of wireless connection). The worker's current context may change quickly, which is especially relevant for MCS applications with non-spatial tasks since workers may complete such tasks anytime and anywhere. We summarize such factors again under the term *context*.

Formally, a worker's current context is modeled as follows. We denote the number of personal context dimensions of a worker $i \in \mathcal{W}$ by X_i and we denote the X_i-dimensional personal context space by $\mathcal{X}_i$. The context space $\mathcal{X}_i$ is assumed to be bounded and can hence be set to $\mathcal{X}_i := [0,1]^{X_i}$ without loss of generality. Here, we allow each worker $i \in \mathcal{W}$ to have an individual personal context space $\mathcal{X}_i$ since each worker may allow the MCS application access to an individual set of context dimensions (e.g., the worker allows access to a certain set of sensors of the mobile device that are used to derive her/his context). In each of the X_i context dimensions, the context corresponds to a value between $[0,1]$ such that the overall personal worker context is hence a vector in $[0,1]^{X_i}$. In practice, the context information may be derived using data collection resources from the mobile devices as indicated in Figure 5.1, e.g., based on sensor readings. Moreover, a pre-processing of the collected data may be needed in order to bring the feature information into the correct format.

We further call $\mathcal{X}_i \times \mathcal{C} = [0,1]^{X_i} \times [0,1]^C \equiv [0,1]^{D_i}$ the *joint (personal and task) context space* of worker $i \in \mathcal{W}$, where $D_i := X_i + C$ is the dimension of this joint context space. The *joint (personal and task) context* of worker i is hence a vector in $[0,1]^{D_i}$. The reason for considering the joint context is that the performance of a worker on a specific task at a specific point in time may depend jointly on the current personal context and on the task context.

Based on the above notation, for a task t, the task context is denoted by $\mathbf{c}_t \in [0,1]^C$, the personal context of a worker $i \in \mathcal{W}_t$ at the arrival time of task t is denoted by $\mathbf{x}_{t,i} \in \mathcal{X}_i$ and the joint (personal and task) context of worker $i \in \mathcal{W}_t$ is given by the concatenation $(\mathbf{x}_{t,i}, \mathbf{c}_t) \in \mathcal{X}_i \times \mathcal{C}$. We do not make any assumptions on the nature of the task context arrival process other than that the sequence $\{\mathbf{c}_t\}_{t=1,\dots,T}$ is not influenced by decisions about worker selection taken by the MCSP. Moreover, we do not make any assumptions on the nature of the personal context arrival process of any worker $i \in \mathcal{W}$ other than that the sequence $\{\mathbf{x}_{t,i}\}_{\{t:i\in\mathcal{W}_t\}}$ is not influenced by decisions about worker selection taken by the MCSP.

5.3.4 Model of Context-Specific Worker Performance

Next, a model of worker performance in dependence of the joint personal and task context is proposed. The performance of a worker is a function of both (i) the worker's willingness to accept a task and (ii) the worker's quality in completing a task. We assume that a worker's quality can take values in a range $[q_{\min}, q_{\max}] \subseteq \mathbb{R}_{0,+}$. Both the willingness to accept a task and the quality may depend on the worker's current personal context and on the task context. Let $p_i(\mathbf{x}, \mathbf{c})$ denote the performance of worker $i \in \mathcal{W}$ with current personal context $\mathbf{x} \in \mathcal{X}_i$ for a task with task context $\mathbf{c} \in \mathcal{C}$. The performance can be decomposed into (i) worker i's decision $d_i(\mathbf{x}, \mathbf{c})$ to accept ($d_i(\mathbf{x}, \mathbf{c}) = 1$) or reject ($d_i(\mathbf{x}, \mathbf{c}) = 0$) the task and, in case the worker accepts the task, also on (ii) worker i's quality $q_i(\mathbf{x}, \mathbf{c})$ when completing the task. Hence, we can write

$$p_i(\mathbf{x}, \mathbf{c}) := q_i(\mathbf{x}, \mathbf{c}) d_i(\mathbf{x}, \mathbf{c}). \tag{5.1}$$

The performance $p_i(\mathbf{x}, \mathbf{c})$ is a random variable whose distribution depends on the distributions of the random variables $d_i(\mathbf{x}, \mathbf{c})$ and $q_i(\mathbf{x}, \mathbf{c})$. Since the decision $d_i(\mathbf{x}, \mathbf{c})$ is binary, it is drawn from a Bernoulli distribution with unknown parameter $\pi_i(\mathbf{x}, \mathbf{c}) \in [0, 1]$. Here, $\pi_i(\mathbf{x}, \mathbf{c}) = \mathbb{E}[d_i(\mathbf{x}, \mathbf{c})]$ represents worker i's acceptance rate given the joint context $(\mathbf{x}, \mathbf{c})$. The quality $q_i(\mathbf{x}, \mathbf{c})$ is a random variable with unknown distribution and we denote its expected value conditioned on $d_i(\mathbf{x}, \mathbf{c}) = 1$ (i.e., task acceptance) by $\nu_i(\mathbf{x}, \mathbf{c}) := \mathbb{E}[q_i(\mathbf{x}, \mathbf{c}) | d_i(\mathbf{x}, \mathbf{c}) = 1]$. Hence, $\nu_i(\mathbf{x}, \mathbf{c})$ represents the average quality of worker i with personal context $\mathbf{x}$ when completing a task of context $\mathbf{c}$ after accepting it. Therefore, the performance $p_i(\mathbf{x}, \mathbf{c})$ of worker $i \in \mathcal{W}$ given the joint context $(\mathbf{x}, \mathbf{c})$ has unknown distribution, takes values in $[0, q_{\max}]$ and its expected value satisfies

$$\mathbb{E}[p_i(\mathbf{x}, \mathbf{c})] = \theta_i(\mathbf{x}, \mathbf{c}), \tag{5.2}$$

where $\theta_i(\mathbf{x}, \mathbf{c}) := \pi_i(\mathbf{x}, \mathbf{c}) \nu_i(\mathbf{x}, \mathbf{c})$.

Based on the above notation, for a task t, given the joint personal and task context $(\mathbf{x}_{t,i}, \mathbf{c}_t) \in \mathcal{X}_i \times \mathcal{C}$ of a worker $i \in \mathcal{W}_t$, the random variable describing the performance of this worker on this task is given by $p_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$ and its expected value is given by $\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$. We assume that for any task $t \in \{1, ..., T\}$, the random variables $\{p_i(\mathbf{x}_{t,i}, \mathbf{c}_t)\}_{i \in \mathcal{W}_t}$ are independent of each other and each random variable $p_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$ is independent of past decisions about worker selection taken by the MCSP and of previous worker performances. Moreover, by $p_i(\mathbf{x}_{t,i}, \mathbf{c}_t, t)$, we denote the actual instantaneous performance of a worker $i \in \mathcal{W}_t$ that has been requested to complete task t, i.e., the realization of the random variable $p_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$ for task t.

5.3.5 Architecture of Decision Making

We propose to use a *hierarchical* architecture of decision making, cf. Section 2.2.3, by splitting the decision making on the one hand, and the information collection and learning, on the other hand, between several entities. We use a hierarchical architecture since the workers' personal contexts should be kept locally in order to keep the communication overhead small and to protect the workers' privacy, but at the same time, the worker selection should be centrally coordinated in order to ensure that the best workers are selected. In the proposed model, the MCSP is responsible for worker selection. For selecting suitable workers, context-specific worker performance needs to be taken into account, however, personal worker context should not be shared with the MCSP. Therefore, we propose that in the mobile device of each worker, a software called *local controller* (LC) is installed. We denote by LC i the LC of worker i. Depending on the requirements of the specific MCS application concerning communication overhead and the requirements of the workers concerning privacy, the LCs may be owned by different parties. Either the MCSP may own the LCs and in this case, the LCs could be part of the software of the MCS application. Alternatively, the LCs may also be included in a separate software that could be owned by either the workers or by a trusted third party [GWG+16, TGFS17]. The proposed model and algorithm are agnostic to the specific owners of the LCs, as long as each LC has access to its corresponding worker's personal context. Moreover, we propose that an LC has the following functionality:

- The LC can reach its worker via the user interface of the MCS application.
- The LC can communicate with the MCSP and, if needed, also with task owners.
- The LC can perform low complexity processing, such as storing, comparing and updating variables and performing simple arithmetic operations on them.

Using this functionality, in the algorithm proposed in this chapter, the LC of a worker is responsible for regularly executing the following tasks sequentially:

(i) Observe the worker's context

(ii) Estimate the worker's context-specific performance and send it to the MCSP

(iii) Observe the worker's instantaneous performance when the worker was requested by the MCSP to complete a task

In this way, over time, the LC learns its worker's context-specific expected performance based on observations and the MCSP is enabled to select an appropriate subset of workers for each task.

5.3.6 Action Model

In the proposed model, the MCSP selects workers for each task. This is formalized in terms of actions taken by the MCSP as follows. We introduce a binary variable $y_{t,i}$ for each task $t \in \{1, ..., T\}$ and each worker $i \in \mathcal{W}_t$, where

$$y_{t,i} := \begin{cases} 1, & \text{if worker } i \text{ is requested to complete task } t, \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

As described in Section 5.3.2, for task t, based on the budget b_t and the price e_t per worker, the MCSP may select $m_t := \lfloor \frac{b_t}{e_t} \rfloor \in \{1, ..., W\}$ workers. Therefore, the following constraints have to hold:

$$\sum_{i \in \mathcal{W}_t} y_{t,i} \leq m_t \ \forall t = 1, ..., T. \tag{5.4}$$

We denote the set of workers that the MCSP selects and requests to complete task t by

$$\mathcal{S}_t := \{i \in \mathcal{W}_t : y_{t,i} = 1\}. \tag{5.5}$$

5.4 Problem Formulation

5.4.1 Formal Problem Statement

In this section, based on the models presented in Section 5.3, we formulate the problem of context-aware worker selection for maximizing the worker performance in MCS applications with non-spatial tasks to be solved in hierarchical fashion by the MCSP and the LCs. As stated above, we assume that tasks $t = 1, ..., T$ arrive sequentially. Consider now an *arbitrary sequence of T task and worker arrivals*, i.e., consider a sequence of tasks $t = 1, ..., T$ with arbitrary task budgets $\{b_t\}_{t=1,...,T}$, arbitrary task contexts $\{\mathbf{c}_t\}_{t=1,...,T}$, arbitrary worker availability $\{\mathcal{W}_t\}_{t=1,...,T}$ and arbitrary worker contexts $\{\mathbf{x}_{t,i}\}_{i \in \mathcal{W}_t, t=1,...,T}$. The goal of the system of MCSP and LCs is to select workers for each task in such a way that the expected cumulative worker performance up

to task T is maximized. Based on the action model in Section 5.3.6, the problem of selecting, for each task, a subset of workers which maximizes the sum of expected performances given the task budget is given by

$$\begin{aligned} \max \ & \sum_{t=1}^{T} \sum_{i \in \mathcal{W}_t} \theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t) y_{t,i} \\ \text{s.t.} \ & \sum_{i \in \mathcal{W}_t} y_{t,i} \leq m_t \ \forall t = 1, ..., T \\ & y_{t,i} \in \{0, 1\} \ \ \forall i \in \mathcal{W}_t, \ \forall t = 1, ..., T. \end{aligned} \tag{5.6}$$

with $y_{t,i}$ of (5.3), $\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$ as defined in Section 5.3.4 and the constraints from (5.4). Problem (5.6) includes the expected context-specific worker performances $\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$, i.e., the expected performances of available workers as functions of their joint personal and task contexts.

5.4.2 Oracle Solution

First, we analyze Problem (5.6) under the assumption that there would be an entity that *had* a priori knowledge about context-specific worker performances and access to the current personal worker contexts. Hence, only in this section, suppose that there exists an entity which (i) is an *omniscient oracle*, knowing the expected performance of each worker under each joint personal and task context *a priori*, and that this entity (ii) is *centrally* informed about the current personal contexts of all available workers for each arriving task.

For such an entity, Problem (5.6) corresponds to an ILP problem, cf. Section 2.3.2.2. As the sub-problems in Problem (5.6) for the different tasks are not coupled, Problem (5.6) can be decoupled into T independent sub-problems, one for each arriving task. For a task t, if fewer workers are available than required, i.e., $W_t \leq m_t$, the trivial optimal solution of the sub-problem associated to task t is to request all available workers to complete the task. In contrast, if for a task t, $W_t > m_t$ holds, the sub-problem associated to task t corresponds to a knapsack problem, cf. Section 2.3.2.3, with a knapsack of capacity m_t and with $W_t = |\mathcal{W}_t|$ items, where item $i \in \mathcal{W}_t$ has a unit weight and a non-negative profit $\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$. Due to the unit weights, the sub-problem in this case actually is a special case of the knapsack problem that may be solved efficiently. Indeed, the optimal solution of the sub-problem can be easily computed in a running time of at most $O(W \log(W))$ as follows. The optimal solution is given by ranking the available workers in $\mathcal{W}_t$ according to their context-specific expected performances and by selecting the m_t highest ranked workers.

For a task $t \in \{1, ..., T\}$, we denote an optimal subset of workers to select for the task by $\mathcal{S}_t^* := \{s_{t,1}^*, ..., s_{t,\min\{m_t, W_t\}}^*\}$. Formally, these workers satisfy

$$s_{t,j}^* \in \underset{i \in \mathcal{W}_t \setminus \bigcup_{k=1}^{j-1}\{s_{t,k}^*\}}{\operatorname{argmax}} \theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t) \text{ for } j = 1, ..., \min\{m_t, W_t\}, \tag{5.7}$$

where $\bigcup_{k=1}^{0}\{s_{t,k}^*\} := \emptyset$. Note that several workers may have the same expected performance and hence the optimal set of workers may not be unique, which is also captured here. Moreover, note that an optimal set $\mathcal{S}_t^*$ of workers for task t depends on the task budget b_t, task context $\mathbf{c}_t$, price e_t, the set $\mathcal{W}_t$ of available workers and their personal contexts $\{\mathbf{x}_{t,i}\}_{i \in \mathcal{W}_t}$, but we write $\mathcal{S}_t^*$ instead of $\mathcal{S}_t^*(b_t, \mathbf{c}_t, e_t, \mathcal{W}_t, \{\mathbf{x}_{t,i}\}_{i \in \mathcal{W}_t})$ for brevity. Let

$$\mathcal{S}^* := \{\mathcal{S}_t^*\}_{t=1,...,T} \tag{5.8}$$

be the collection of optimal subsets of workers for the collection $\{1, ..., T\}$ of tasks. We call this collection the *centralized oracle solution*, since it requires an entity with a priori knowledge about expected context-specific worker performances and with access to personal worker contexts to make optimal decisions.

5.4.3 Contextual Multi-Armed Bandit Formulation

Now, we characterize Problem (5.6) under the conditions actually faced by the MCSP and LCs. Namely, the set of MCSP and LC do not have a priori knowledge about expected performances, and the workers' personal contexts are only locally available in each mobile device, but may not be shared with the MCSP.

If for an arriving task t, fewer workers are available than required, i.e., $W_t \leq m_t$, by simply requesting all available workers (i.e., $\mathcal{S}_t = \mathcal{W}_t$) to complete the task, the MCSP automatically selects the optimal subset of workers. Otherwise, if $W_t > m_t$ holds for an arriving task t, the MCSP cannot simply solve the sub-problem for task t appearing in Problem (5.6) like the centralized oracle. This is because on the one hand, it does not know the expected performances $\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t)$ and on the other hand, the MCSP cannot access the workers' personal contexts. Hence, in this case, a *machine-learning-based* approach, cf. Section 2.3.1, is needed since the system of MCSP and LCs can only learn the workers' performances by selecting different workers over time and observing their instantaneous performances.

Considering the problem statement in Section 5.4.1, under the conditions actually faced by the MCSP and LCs, Problem (5.6) can be understood as a contextual MAB problem

as follows, cf. Section 2.3.3.4. The MCSP and LC can be understood as a set of agents, where one of them (the MCSP) needs to sequentially select from a set of actions. In the considered MCS problem, the set of actions is given by the set $\mathcal{W}$ of workers. There is a sequence of tasks $t = 1, ..., T$ that corresponds to a sequence of rounds faced by the agents. At the arrival of a task t, only a subset $\mathcal{W}_t \subseteq \mathcal{W}$ of workers may be available and hence the set of actions may be different in each round. For each arriving task, the following events happen sequentially. First, the MCSP receives the task and especially observes the task context $\mathbf{c}_t$. Moreover, the LC of each available worker $i \in \mathcal{W}_t$ observes its worker's personal context $\mathbf{x}_{t,i}$. This corresponds to several contexts revealed to the agents in the beginning of a round. Secondly, the MCSP selects a subset of $\min\{m_t, W_t\}$ workers from set $\mathcal{W}_t$ and requests them to complete the task. This corresponds to an agent selecting a subset of available actions. Thirdly, each LC of a requested worker observes the instantaneous performance of the worker. This corresponds to the agents receiving a reward for each selected action. Taking into account the assumptions about the arrival processes of the tasks, workers and their performances in Sections 5.3.2 – 5.3.4, Problem (5.6) corresponds to a contextual MAB problem with a similar model as the one presented in Section 2.3.3.4. The main differences between these two models are as follows:

- In Problem (5.6), the agent may select several actions per round instead of only one and the number of actions to be selected may be different in each round. Therefore, formally, Problem (5.6) is a contextual *combinatorial* MAB problem, cf. Section 2.3.3.2. However, since neither the objective function nor the constraints in Problem (5.6) are combinatorial, Problem (4.4) is more accurately a contextual MAB problem with several action selections per round, but not of combinatorial nature.

- In Problem (5.6), actions may be unavailable in arbitrary rounds, whereas in the model in Section 2.3.3.4, actions are always available. Therefore, Problem (5.6) is a contextual MAB problem with *sleeping arms*, cf. Section 2.3.3.2.

- Instead of one agent as in the model in Section 2.3.3.4, Problem (5.6) has to be solved cooperatively by several agents, where one *coordinating agent* (i.e., the MCSP) selects a subset of actions in each round based on the estimates of a set of *learning agents* (i.e., the LCs), where each learning agent observes the context of one particular action and learns the rewards of this action.

Consequently, a coordination mechanism between the MCSP and LCs needs to be designed in order to enable the LCs to learn their workers' context-specific performances

over time and to enable the MCSP to select suitable workers for each task to maximize the worker performance on this task given its task budget. Specifically, over time, the system of MCSP and LCs has to use a suitable trade-off between *exploration* and *exploitation*, by, on the one hand, selecting workers about whose performance only little information is available and, on the other hand, selecting workers who are likely to have high performance. For each arriving task, the selection of workers depends on the history of previously selected workers and the corresponding observed performances. Since observing worker performance requires quality assessments that may be costly, the number of performance observations should be limited in order to keep the cost for quality assessment low. An algorithm which maps the history of previously selected workers and observed performances to the next selections of workers is called a *learning algorithm*. The performance of such a learning algorithm can be evaluated by comparing its loss with respect to the centralized oracle solution given in (5.8) in terms of the achieved cumulative worker performance. Formally, for an arbitrary sequence of T task and worker arrivals, the *regret* of learning with respect to the centralized oracle solution is given by

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{\min\{m_t,W_t\}}\Big(p_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}},\mathbf{c}_t,t) - p_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}},\mathbf{c}_t,t)\Big)\right]. \tag{5.9}$$

where $p_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}},\mathbf{c}_t,t)$ denotes the instantaneous performance of the selected worker $s_{t,j} \in \mathcal{S}_t$, $j \in \{1,...,\min\{m_t,W_t\}\}$, with personal worker context vector $\mathbf{x}_{t,i}$ for task t with task context $\mathbf{c_t}$. Here, the expectation is taken with respect to the selections $\{\mathcal{S}_t\}_{t=1,...,T}$ made by the learning algorithm and the randomness of the workers' performances.

Equivalently, one can write the regret $R(T)$ as

$$R(T) = \sum_{t=1}^{T}\sum_{j=1}^{\min\{m_t,W_t\}}\Big(\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}},\mathbf{c}_t) - \mathrm{E}[\theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}},\mathbf{c}_t)]\Big). \tag{5.10}$$

5.5 Proposed Algorithm

Based on the formulation as a contextual MAB problem given in Section 5.4.3, we propose a context-aware hierarchical online learning algorithm for worker selection in MCS. The algorithm is based on the assumption that a worker's expected performance is similar in similar joint personal and task contexts. Therefore, by observing the task context, a worker's personal context and her/his performance when requested to

complete a task, the worker's context-specific expected performances can be learned and exploited for future worker selection.

The proposed algorithm is based on the contextual MAB algorithms in [TvdS15a, TZvdS14, MAvK16, MAvK17] and extends these works as follows:

- While in [TvdS15a, TZvdS14, MAvK16, MAvK17], a learning agent observes a set of contexts and selects a subset of actions based on these contexts, the proposed algorithm is decoupled to several learning agents, each observing the context of one particular action and learning the rewards of this action, and a coordinating agent, which selects a subset of actions based on the learning agents' estimates. In the considered MCS problem, an action corresponds to a worker, the learning agents correspond to the LCs which learn the performances of their workers, and the coordinating agent corresponds to the MCSP, which selects workers based on the performance estimates from the LC.
- While in [TvdS15a, TZvdS14, MAvK16, MAvK17], the same number of actions is selected per round, the proposed algorithm allows different numbers of actions to be selected per round. In the considered MCS problem, this corresponds to allowing different required numbers of workers for different tasks. Hence, in contrast to [TvdS15a, TZvdS14, MAvK16, MAvK17], the learning speed of the proposed algorithm is affected by the arrival process of the numbers of actions to be selected.
- While in [TvdS15a, TZvdS14, MAvK16, MAvK17], each action has the same context space, the proposed algorithm allows each action to have an individual context space of an individual dimension. In the considered MCS problem, this corresponds to allowing workers to give access to individual sets of context dimensions. Therefore, in contrast to [TvdS15a, TZvdS14, MAvK16, MAvK17], the granularity of learning may be different for different actions.
- Finally, while in [TvdS15a, TZvdS14, MAvK16, MAvK17], all actions are available in any round, the proposed algorithm allows actions to be unavailable in arbitrary rounds. In the considered MCS problem, this corresponds to allowing that workers may be unavailable. Hence, in contrast to [TvdS15a, TZvdS14, MAvK16, MAvK17], the best subset of actions in a certain round depends on the specific set of available actions in this round.

We call the proposed algorithm the *hierarchical context-aware learning* (HCL) algorithm. Fig. 5.2 shows an overview of the main steps of HCL for a task $t \in \{1, ..., T\}$

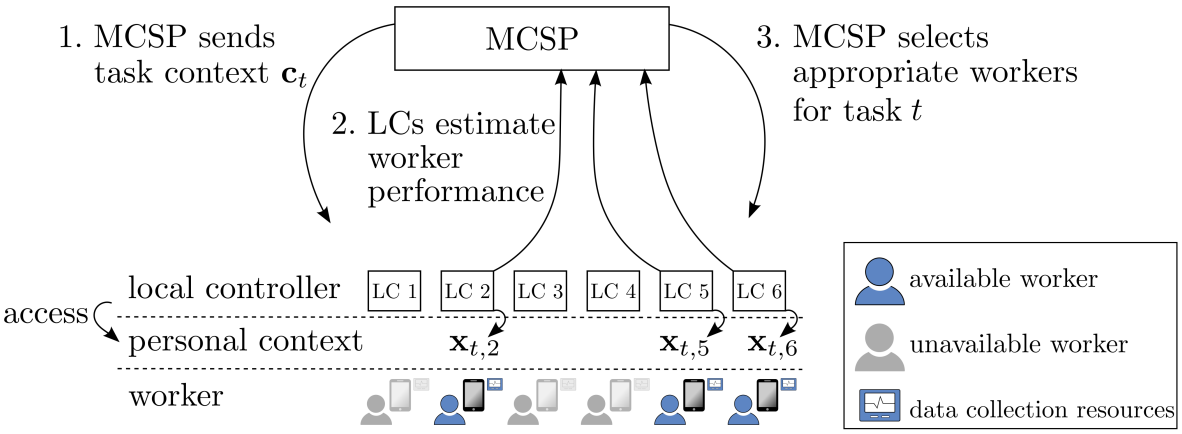


Figure 5.2. Overview of operation of HCL algorithm for task t.

in an exemplary MCS application with $W = 6$ workers. A short summary of HCL is given next. When a task arrives at the MCSP, the MCSP broadcasts the corresponding task context to the LCs. Upon receiving information about a task, the LC of an available worker first observes its worker's personal context. Then, on the one hand, if its worker's performance has been observed sufficiently often before, given the current joint personal and task context, the LC relies on previous performance observations to estimate its worker's performance and sends a performance estimate to the MCSP. On the other hand, if its worker's performance has not been observed sufficiently often before, the LC informs the MCSP that its worker has to be explored. Subsequently, based on the messages received from the LCs of available workers, the MCSP selects a subset of workers. The LC of a worker selected by the MCSP then requests its worker to complete the task and observes whether or not the worker accepts the task. In case the worker was selected for exploration purposes and then accepts the task, the LC additionally observes the quality of the completed task. How exactly the quality of the task is observed, depends on the type of quality assessment used by the MCS application. For example, the LC may get a quality rating from the task owner or the LC may generate an automatic quality assessment using either local software or the resources of a cloud. The reason for only making a quality assessment when a worker was selected for exploration purposes is that quality assessment may be costly and therefore, HCL keeps the number of quality assessments low. Clearly, if quality assessment in the MCS application is cheap, HCL can be adapted to always observe worker quality, which may increase the learning speed.

In HCL, a worker's personal contexts and performance observations in terms of task acceptance and quality are only locally stored by the LC. Thereby, (i) personal context is kept locally, (ii) the space required for storing worker information at the MCSP is kept low, (iii) if necessary, task completion and result transmission may be directly handled

between the LC and the task owner, (iv) workers are assigned to tasks which they are interested in and at which they are good, but without the need to share their context information, and (v) even though an LC has to keep track of its worker's personal contexts and performance observations, the computation and memory overhead for each LC is small.

Next, HCL is discussed in more detail, starting with the operation of the LCs. The pseudocode of HCL for LC i is given in Algorithm 5.1. First, for synchronization purposes, LC i receives the finite number T of tasks to be considered, the task context space $\mathcal{C}$ and its dimension C from the MCSP. In addition, LC i checks to which context dimensions of worker i it has access in order to define the personal context space $\mathcal{X}_i$ and its dimension X_i. Thereafter, LC i sets the joint context space to $\mathcal{X}_i \times \mathcal{C}$ with size $D_i = X_i + C$. Moreover, LC i needs to set a parameter $h_{T,i} \in \mathbb{N}$ and a control function $K_i : \{1, ..., T\} \to \mathbb{R}_+$ during initialization, which are both described below. Then, LC i initializes a uniform partition $\mathcal{Q}_{T,i}$ of worker i's joint context space $[0, 1]^{D_i}$. The partition consists of $(h_{T,i})^{D_i}$ D_i-dimensional hypercubes of equal size $\frac{1}{h_{T,i}} \times \ldots \times \frac{1}{h_{T,i}}$. Hence, the granularity of the partition of the context space is determined by the parameter $h_{T,i}$. Additionally, LC i initializes a counter $N_{i,q}(t)$ for each hypercube $q \in \mathcal{Q}_{T,i}$. The counter $N_{i,q}(t)$ represents the number of times before (i.e., up to, but not including) task t, in which worker i was selected to complete a task for exploration purposes when her/his joint context belonged to hypercube q. Finally, for each hypercube $q \in \mathcal{Q}_{T,i}$, LC i initializes the estimate $\hat{\theta}_{i,q}(t)$, which represents the estimated performance of worker i for contexts in hypercube q at the arrival of task t.

For each of the arriving tasks $t = 1, ..., T$, LC i executes the following steps. LC i only takes actions for an arriving task t, if its worker i is currently available, i.e., if $i \in \mathcal{W}_t$ holds. This is because if a worker is unavailable, it may mean that she/he is offline. Therefore, we consider the LC to only take actions if its worker is available. If this is the case, LC i first receives the task context $\mathbf{c}_t$ from the MCSP. Additionally, LC i observes worker i's current personal context $\mathbf{x}_{t,i}$ and determines the hypercube from $\mathcal{Q}_{T,i}$ to which the joint context $(\mathbf{x}_{t,i}, \mathbf{c}_t)$ belongs. If there are multiple such hypercubes, one of them is randomly selected. We denote this hypercube by $q_{t,i} \in \mathcal{Q}_{T,i}$. It satisfies $(\mathbf{x}_{t,i}, \mathbf{c}_t) \in q_{t,i}$. Subsequently, LC i checks if worker i has not been selected sufficiently often before when worker i's joint personal and task context belonged to hypercube $q_{t,i}$. For this purpose, LC i compares the counter $N_{i,q_{t,i}}(t)$ with $K_i(t)$, where $K_i : \{1, ..., T\} \to \mathbb{R}_+$ is a deterministic, monotonically increasing control function, set in the beginning of the algorithm. On the one hand, if worker i has been selected sufficiently often before ($N_{i,q_{t,i}}(t) > K_i(t)$), LC i relies on the estimated performance $\hat{\theta}_{i,q_{t,i}}(t)$, and sends it to the MCSP. On the other hand, if worker i has not been selected sufficiently often before ($N_{i,q_{t,i}}(t) \leq K_i(t)$), LC i sends an "explore"

Algorithm 5.1 HCL@LC: Local Controller i of Worker i.

Receive input from MCSP: T, $\mathcal{C}$, C
Receive input from worker i: $\mathcal{X}_i$, X_i
Set joint context space $\mathcal{X}_i \times \mathcal{C}$, set $D_i = X_i + C$
Set parameter $h_{T,i} \in \mathbb{N}$ and control function $K_i : \{1, ..., T\} \to \mathbb{R}_+$
Initialize context partition: Create partition $\mathcal{Q}_{T,i}$ of $[0,1]^{D_i}$ into $(h_{T,i})^{D_i}$ hypercubes of identical size
Initialize counters: For all $q \in \mathcal{Q}_{T,i}$, set $N_{i,q} = 0$
Initialize estimated performance: For all $q \in \mathcal{Q}_{T,i}$, set $\hat{\theta}_{i,q} = 0$
for each $t = 1, ..., T$ **do**
    if $i \in \mathcal{W}_t$ **then**
        Receive task context $\mathbf{c}_t$
        Observe worker i's personal context $\mathbf{x}_{t,i}$
        Find the set $q_{t,i} \in \mathcal{Q}_{T,i}$ such that $(\mathbf{x}_{t,i}, \mathbf{c}_t) \in q_{t,i}$
        if $N_{i,q_{t,i}} > K_i(t)$ **then**
            Send message$_i$:= $\hat{\theta}_{i,q_{t,i}}$ to MCSP
        else
            Send message$_i$:= "explore" to MCSP
        end if
        Wait for MCSP's worker selection
        if MCSP selects worker i **then**
            Give task context $\mathbf{c}_t$ to worker i
            Request worker i to complete task t
            Observe worker i's decision d
            if message$_i$ == "explore" **then**
                if $d == 1$ **then**
                    Observe worker i's quality q, set $p := q$
                else
                    Set $p := 0$
                end if
                $\hat{\theta}_{i,q_{t,i}} = \frac{\hat{\theta}_{i,q_{t,i}} N_{i,q_{t,i}} + p}{N_{i,q_{t,i}} + 1}$
                $N_{i,q_{t,i}} = N_{i,q_{t,i}} + 1$
            end if
        end if
    end if
end for

message to the MCSP. The control function $K_i(t)$ is hence used to decide whether a worker should be selected for exploration purposes (to achieve reliable estimates) or whether the worker's performance estimates are already reliable and can be exploited. Since the trade-off between exploration and exploitation is hence determined by the control function, the choice of the control function is crucial to ensure a good result of the learning algorithm. A suitable choice of the control function will be proposed in Section 5.6.1.

Thereafter, LC i waits for the MCSP to take care of the worker selection. On the one hand, if the MCSP does not select worker i, LC i does not take further actions. On the other hand, if the MCSP selects worker i, LC i hands the task context information $\mathbf{c}_t$ to worker i via the user interface of the MCS application and requests worker i to complete the task. Then, LC i observes whether or not worker i accepts the task. If worker i was selected for exploration purposes, LC i makes an additional update of a counter and of an estimate. For this purpose, if worker i accepted the task, LC i additionally observes worker i's quality in completing the task (e.g., by requesting a quality rating from the task owner or by generating an automatic quality assessment) and sets the observed performance to the observed quality. If worker i declined the task, LC i sets the observed performance to 0. Then, based on the observed performance, LC i computes the estimated performance $\hat{\theta}_{i,q_{t,i}}(t+1)$ for hypercube $q_{t,i}$ and the counter $N_{i,q_{t,i}}(t+1)$. Note that in Algorithm 5.1, the argument t is omitted from counters $N_{i,q}(t)$ and estimates $\hat{\theta}_{i,q}(t)$ since it is not necessary to store their respective previous values.

By definition of HCL, the estimated performance $\hat{\theta}_{i,q}(t)$ corresponds to the product of (i) the relative frequency with which worker i accepted tasks when the joint context belonged to hypercube q and (ii) the average quality in completing these tasks. Formally, the estimate $\hat{\theta}_{i,q}(t)$ is computed as follows. Let $\mathcal{E}_{i,q}(t)$ be the set of observed performances of worker i before task t when worker i was selected for a task and the joint context was in hypercube q. If before task t, worker i's performance has never been observed before for a joint context in hypercube q, we have $\mathcal{E}_{i,q}(t) = \emptyset$ and $\hat{\theta}_{i,q}(t) := 0$. Otherwise, the estimated performance is given by $\hat{\theta}_{i,q}(t) := \frac{1}{|\mathcal{E}_{i,q}(t)|} \sum_{p \in \mathcal{E}_{i,q}(t)} p$. Note that the set $\mathcal{E}_{i,q}(t)$ does not appear in HCL since the estimated performance $\hat{\theta}_{i,q}(t)$ can be computed based on $\hat{\theta}_{i,q}(t-1)$, $N_{i,q}(t-1)$ and on the performance for task $t-1$.

Next, we discuss the operation of the MCSP. The pseudocode of HCL for the MCSP is given in Algorithm 5.2. First, for synchronization purposes, the MCSP informs the LCs about the finite number T of tasks to be considered, the task context space $\mathcal{C}$ and its dimension C. Then, for an arriving task t with budget and task context given by $(b_t, \mathbf{c}_t)$, the MCSP computes the maximum required number m_t of workers, based

Algorithm 5.2 HCL@MCSP: Worker Selection at MCSP.

Send input to LCs: T, $\mathcal{C}$, C
for each $t = 1, ..., T$ **do**
  Receive task t with budget and task context $(b_t, \mathbf{c}_t)$
  Compute $m_t = \lfloor \frac{b_t}{e_t} \rfloor$
  Set $\mathcal{W}_t = \emptyset$
  Set $\mathcal{W}_t^{\text{ue}} = \emptyset$
  Broadcast task context $\mathbf{c}_t$
  for each $i = 1, ..., W$ **do**
    if Receive message$_i$ from LC i **then**
      $\mathcal{W}_t = \mathcal{W}_t \cup \{i\}$
      if message$_i$ == "explore" **then**
        $\mathcal{W}_t^{\text{ue}} = \mathcal{W}_t^{\text{ue}} \cup \{i\}$
      end if
    end if
  end for
  Compute $W_t = |\mathcal{W}_t|$
  if $W_t \leq m_t$ **then** ▷ SELECT ALL
    Select all W_t workers from $\mathcal{W}_t$
  else
    Compute $W_{\text{ue},t} = |\mathcal{W}_t^{\text{ue}}|$
    if $W_{\text{ue},t} == 0$ **then** ▷ EXPLOITATION
      Rank workers in $\mathcal{W}_t$ according to estimates from $(\text{message}_i)_{i \in \mathcal{W}_t}$
      Select the m_t highest ranked workers
    else ▷ EXPLORATION
      if $W_{\text{ue},t} \geq m_t$ **then**
        Select m_t workers randomly from $\mathcal{W}_t^{\text{ue}}$
      else
        Select the $W_{\text{ue},t}$ workers from $\mathcal{W}_t^{\text{ue}}$
        Rank workers in $\mathcal{W}_t \setminus \mathcal{W}_t^{\text{ue}}$ according to estimates from $(\text{message}_i)_{i \in \mathcal{W}_t \setminus \mathcal{W}_t^{\text{ue}}}$
        Select the $(m_t - W_{\text{ue},t})$ highest ranked workers
      end if
    end if
  end if
  Inform LCs of selected workers
end for

on the budget b_t and the corresponding price e_t per worker. Moreover, the MCSP initializes two sets. First, the set $\mathcal{W}_t$ represents the set of available workers when task t arrives. Secondly, the set $\mathcal{W}_t^{\text{ue}}$ is the so-called *set of under-explored workers*, containing all available workers which have not been selected sufficiently often before given their respective current joint personal and task context. After broadcasting the task context $\mathbf{c}_t$, the MCSP waits for messages from the LCs. When the MCSP receives a message from an LC, it adds the corresponding worker to the set $\mathcal{W}_t$ of available workers. In addition, the MCSP checks whether a received message is an "explore" message. If this is the case, the MCSP additionally adds the corresponding worker to the set $\mathcal{W}_t^{\text{ue}}$ of under-explored workers. Note that by the definitions of Algorithms 5.1 and 5.2, the set of under-explored workers is hence given by

$$\mathcal{W}_t^{\text{ue}} = \{i \in \mathcal{W}_t : N_{i,q_{t,i}}(t) \leq K_i(t)\}. \tag{5.11}$$

Next, the MCSP computes the number W_t of available workers. If $W_t \leq m_t$ holds, i.e., at most the required number of workers is available, the MCSP enters a *select-all-workers phase* and selects all available workers to complete the task. Otherwise, the MCSP continues by calculating the number $W_{\text{ue},t} := |\mathcal{W}_t^{\text{ue}}|$ of under-explored workers. In case there is no under-explored worker, the MCSP enters an *exploitation phase*. It ranks the available workers in $\mathcal{W}_t$ according to their estimated performances, which it received from the respective LCs, and then selects the m_t highest ranked workers. This approach enables the MCSP to make use of context-specific estimated performances without actually observing the workers' personal contexts. In case there are under-explored workers, the MCSP enters an *exploration phase*. These phases are needed such that all LCs are able to update their estimated performances sufficiently often. Depending on the number $W_{\text{ue},t}$ of under-explored workers, two different cases may occur. Either the number $W_{\text{ue},t}$ of under-explored workers is at least m_t, in which case the MCSP selects m_t under-explored workers at random. Or the number $W_{\text{ue},t}$ of under-explored workers is smaller than m_t, in which case the MCSP selects all $W_{\text{ue},t}$ under-explored workers. Since the MCSP may select $m_t - W_{\text{ue},t}$ additional workers, it ranks the available sufficiently-explored workers according to their estimated performances received from the respective LCs, and then additionally selects the $(m_t - W_{\text{ue},t})$ highest ranked workers. Hence, when the number of under-explored workers is small, additional exploitation is carried out in exploration phases.

After worker selection, the MCSP informs the LCs of selected workers that their workers should be requested to complete the task. Note that since the MCSP does not have to keep track of the workers' decisions, the LCs may handle the contact with the task owner directly (e.g., the task owner may send detailed task instructions directly to an

LC of a selected worker; after task completion, the LC may send the result directly to the task owner).

5.6 Properties of Proposed Algorithm

5.6.1 Upper Bound on Regret

The performance of HCL is evaluated by analyzing its regret with respect to the centralized oracle solution, as defined in (5.10). The theorem given below proves that the regret of HCL is sublinear in T, i.e., there exists $\gamma < 1$ for which $R(T) = O(T^{\gamma})$ holds. Hence, HCL converges to the centralized oracle solution for $T \to \infty$, since $\lim_{T\to\infty} \frac{R(T)}{T} = 0$ holds. The regret bound is derived based on the assumption that under a similar joint personal and task context, a worker's expected performance is also similar. This similarity assumption can be formalized as follows, cf. Section 2.3.3.4.

Assumption 5.1 (Hölder continuity assumption)**.** *There exist $L > 0$, $0 < \alpha \leq 1$ such that*

$$|\theta_i(\mathbf{x}, \mathbf{c}) - \theta_i(\tilde{\mathbf{x}}, \tilde{\mathbf{c}})| \leq L||(\mathbf{x}, \mathbf{c}) - (\tilde{\mathbf{x}}, \tilde{\mathbf{c}})||_{D_i}^{\alpha} \tag{5.12}$$

holds for all joint contexts $(\mathbf{x}, \mathbf{c}), (\tilde{\mathbf{x}}, \tilde{\mathbf{c}}) \in \mathcal{X}_i \times \mathcal{C} \equiv [0, 1]^{D_i}$ of all workers $i \in \mathcal{W}$, where $||\cdot||_{D_i}$ denotes the Euclidean norm in $\mathbb{R}^{D_i}$.

Note that Assumption 5.1 is required to derive the upper bound on the regret. In contrast, HCL may also be applied to data which does not satisfy this similarity assumption. In this case, the regret bound may, however, not hold.

The theorem given below shows that the regret of HCL is sublinear in T.

Theorem 5.1 (Bound for $R(T)$)**.** *Given that Assumption 5.1 holds, when LC i, $i \in \mathcal{W}$, runs Algorithm 5.1 with parameters $K_i(t) = t^{\frac{2\alpha}{3\alpha+D_i}} \log(t)$, $t = 1, ..., T$, and $h_{T,i} = \lceil T^{\frac{1}{3\alpha+D_i}} \rceil$, and the MCSP runs Algorithm 5.2, the regret $R(T)$ is bounded by*

$$\begin{aligned} R(T) \leq q_{\max} W \sum_{i\in\mathcal{W}} 2^{D_i} \left(\log(T) T^{\frac{2\alpha+D_i}{3\alpha+D_i}} + T^{\frac{D_i}{3\alpha+D_i}} \right) \\ + \sum_{i\in\mathcal{W}} \frac{2q_{\max}}{(2\alpha + D_i)/(3\alpha + D_i)} T^{\frac{2\alpha+D_i}{3\alpha+D_i}} + q_{\max} W^2 \frac{\pi^2}{3} + 2 \sum_{i\in\mathcal{W}} L D_i^{\frac{\alpha}{2}} T^{\frac{2\alpha+D_i}{3\alpha+D_i}}. \end{aligned} \tag{5.13}$$

Hence, the leading order of the regret is $O\left(T^{\frac{2\alpha+D_{\max}}{3\alpha+D_{\max}}} \log(T)\right)$, where $D_{\max} := \max_{i\in\mathcal{W}} D_i$.

The proof of Theorem 5.1 is given in Appendix A.7. The idea of the proof is as follows. First, the regret is decomposed into three terms. The three terms represent the regret due to *select-all-workers* phases, *exploration* phases and *exploitation* phases. Each of the three terms is then bounded separately. First, it is shown that the regret due to *select-all-workers* phases is actually always zero, since the MCSP always selects the optimal set of workers in these phases. Bounding the regret due to *exploration* phases works as follows. The loss due to selecting suboptimal workers in exploration phases may be upper-bounded by a constant. Moreover, it can be shown that the number of exploration phases is limited and can be bounded sublinearly in T, given an appropriate choice of the input parameters of the algorithms. Overall, this leads to a sublinear upper bound on the regret due to exploration phases. The idea for bounding the regret due to *exploitation* phases is as follows. First, one distinguishes between two different types of exploitation phases, depending on whether the estimated performance $\hat{\theta}_{i,q_{t,i}}(t)$ of each available worker $i \in \mathcal{W}_t$ in the current hypercube $q_{t,i}$ is "close" to its expected value $\mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]$. Then, for exploitation phases in which the latter holds true, one can show that even if a suboptimal set of workers is selected, the loss cannot be very large, but can in fact be bounded sublinearly in T, given an appropriate choice of input parameters. For the second type of exploitation phases, the loss due to selecting suboptimal workers is upper-bounded by a constant. Moreover, one can show that the number of this type of exploitation phases is limited by a sublinear bound in T given an appropriate choice of input parameters. Overall, this leads to a sublinear upper bound on the regret due to exploitation phases. Then, the overall regret bound follows by setting the appropriate input parameters in the algorithms.

The regret bound given in Theorem 5.1 is sublinear in T, i.e., HCL converges to the optimal worker selection strategy in the sense that when the number T of tasks goes to infinity, the averaged regret $\frac{R(T)}{T}$ diminishes. Moreover, since Theorem 5.1 is applicable for any finite number T of tasks, it characterizes how fast HCL is learning.

5.6.2 Computational Complexity

Here, we analyze the computational complexity of the proposed algorithm for handling one task t. First, the computational complexity of Algorithm 5.1, executed by the LC of a worker $i \in \mathcal{W}$ is analyzed as a function of the dimension D_i of worker i's joint context space. For this purpose, we identify the most computationally expensive procedures in the algorithm. The complexity of line 10 in Algorithm 5.1 does not grow with D_i, and hence the computational complexity of this line is $O(1)$. Observing worker i's personal context has a computational complexity that grows as $O(D_i)$ since

each entry of the context vector has to be accessed once. Finding the hypercube in the partition of the context space to which the joint context belongs, has a complexity that grows as $O(D_i)$. This is because one can give a separate index to each hypercube in each context dimension and then compute the index of the hypercube to which a context vector belongs in each of the D_i context dimension once and independently of the other dimensions. The procedures in the remaining lines of Algorithm 5.1 do not grow with D_i and are hence of order $O(1)$. Overall, the computational complexity of the proposed HCL algorithm for the LC of worker i grows hence as $O(D_i)$, i.e., it grows only linearly with the dimension of the context space.

Next, the computational complexity of Algorithm 5.2 is analyzed as a function of the number W of workers. The complexity of lines 3-7 in Algorithm 5.1 does not grow with W, and hence the computational complexity of these lines is $O(1)$. Determining the set $\mathcal{W}_t$ of available workers and the set $\mathcal{W}_t^{\text{ue}}$ of under-explored workers in lines 8-15 has a complexity that grows as $O(W)$. This is because each worker needs to be considered at most once for each of the two sets. Lines 16 and 20 have a computational complexity that grows at most as $O(W)$ by considering each worker at most once. Lines 17, 21 and 25 do not grow with W and their complexity hence grows as $O(1)$. Selecting workers in either line 18, 23, 26 or 28 and 30 has a computational complexity of $O(W)$ by considering at most each worker once. Ranking the workers in line 22 or line 29 has a computational complexity that grows as $O(W \log W)$ since at most W workers need to be sorted [CLRS09]. Hence, the computational complexity of the proposed HCL algorithm for the MCSP is hence of order $O(W \log W)$, i.e., it has a log-linear complexity with respect to the number of workers.

5.6.3 Local Memory Requirements

We study the required local memory size in the mobile device of a worker when the LC executes Algorithm 5.1. In Algorithm 5.1, LC i stores the counters $N_{i,q}$ and estimates $\hat{\theta}_{i,q}$ for each $q \in \mathcal{Q}_{T,i}$. Using the parameters from Theorem 5.1, the number of hypercubes in the partition $\mathcal{Q}_{T,i}$ is $(h_{T,i})^{D_i} = \lceil T^{\frac{1}{3\alpha+D_i}} \rceil^{D_i} \leq (1 + T^{\frac{1}{3\alpha+D_i}})^{D_i}$. Hence, the number of variables to store in the mobile device of worker i is upper-bounded by $2 \cdot (1 + T^{\frac{1}{3\alpha+D_i}})^{D_i}$. Therefore, the required memory size depends on the number $D_i = X_i + C$ of context dimensions. If a worker allows access to a high number X_i of personal context dimensions and/or the number C of task context dimensions is large, HCL learns the worker's context-specific performance with finer granularity and therefore the assigned tasks are more personalized, but also the required local memory size increases.

5.6.4 Communication Requirements

Next, we deduce the communication requirements of HCL from its main operation steps. First, for a task t, the MCSP broadcasts the task context to the LCs, which is one vector of dimension C (i.e., C scalars), assuming that the broadcast reaches all workers in a single transmission. Subsequently, the LCs of available workers send their workers' estimated performances to the MCSP. This corresponds to W_t scalars to be transmitted, i.e., one scalar sent by each LC of an available worker. Finally, the MCSP informs selected workers about its decision, which corresponds to m_t scalars sent by the MCSP. Therefore, in total, a number $C + W_t + m_t$ of scalars are transmitted for task t. Among these, $C + m_t$ scalars are transmitted by the MCSP and one scalar is transmitted by each mobile device of an available worker.

In order to evaluate whether the hierarchical approach of HCL for decision making induces communication overhead compared to a centralized approach, we now derive the communication requirements of a corresponding centralized approach for context-aware worker selection. In such a centralized approach, for each task, the personal contexts of available workers would be collected in the MCSP, which would then select workers based on the task and personal contexts and finally inform selected workers about its decision. The communication requirements of this centralized approach are hence as follows. First, for a task t, the LC of each available worker i sends the current worker context to the MCSP, which is a vector of dimension D_i (i.e., D_i scalars). In this step, in sum, a number $\sum_{i \in \mathcal{W}_t} D_i$ of scalars are hence transmitted. Then, after worker selection, the MCSP requests selected workers to complete the task, which corresponds to m_t scalars sent by the MCSP. Finally, the MCSP broadcasts the task context to the selected workers, which is one vector of dimension C (i.e., C scalars), assuming that the broadcast reaches all addressed workers in a single transmission. Therefore, in total, a number $\sum_{i \in \mathcal{W}_t} D_i + m_t + C$ of scalars are transmitted for task t. Among these, $C + m_t$ scalars are transmitted by the MCSP and D_i scalars are transmitted by the mobile device of each available worker $i \in \mathcal{W}_t$.

Comparing now HCL with the centralized approach, the mobile device of any worker $i \in \mathcal{W}$ with $D_i > 1$ has to transmit *less* data using HCL than using the centralized approach. Moreover, under the assumption that any broadcast reaches all addressed workers using one single transmission, using HCL instead of the centralized approach reduces the sum communication requirements (for all mobile devices and for the MCSP in sum) by an amount of $\sum_{t=1}^{T} \left(\sum_{i \in \mathcal{W}_t} D_i - W_t \right) \geq 0$ scalars since $D_i \geq 1$ for all $i \in \mathcal{W}$. This shows that (i) the sum communication requirements of using HCL are *at most as high* as that of the centralized approach, and (ii) the more context dimensions the LCs

are allowed to access, the lower are the communication requirements of HCL compared to an equivalent centralized approach.

5.6.5 Worker Quality Assessment Requirements

HCL explicitly takes into account that quality assessments may be costly by only requesting a quality assessment if a worker is selected for exploration purposes. In the sequel, we give an upper bound on the number $A_i(T)$ of quality assessments per worker up to task T.

Corollary 5.1 (Bound for number of quality assessments up to task T)**.** *Given that Assumption 5.1 from Section 5.6.1 holds, when LC i, $i \in \mathcal{W}$, runs Algorithm 5.1 with the parameters given in Theorem 5.1, and the MCSP runs Algorithm 5.2, the number $A_i(T)$ of quality assessments of each worker i up to task T is upper-bounded by*

$$A_i(T) \leq \left(1 + T^{\frac{1}{3\alpha+D_i}}\right)^{D_i} \left(1 + \log(T) T^{\frac{2\alpha}{3\alpha+D_i}}\right). \tag{5.14}$$

The proof of Corollary 5.1 is given in Appendix A.8. The proof of Corollary 5.1 is based on the proof of Theorem 5.1 as follows. Since a quality assessment is only requested in HCL if a worker is selected for exploration purposes, it is sufficient to derive an upper bound on the number of times a worker can at most be selected for exploration purposes. In the proof of Theorem 5.1, it is shown how the number of exploration phases per worker can be upper-bounded sublinearly in T. This upper bound is then used to prove Corollary 5.1. From Corollary 5.1, we see that the number of quality assessments per worker is sublinear in T. Hence, $\lim_{T\to\infty} \frac{A_i(T)}{T} = 0$ holds, so that for $T \to \infty$, the average rate of required quality assessments approaches zero.

5.7 Numerical Results

5.7.1 Simulation Setup

5.7.1.1 Synthetic and Real Data

We evaluate HCL by comparing its performance to several reference algorithms in simulations based on both synthetic as well as real data. The difference between

the two approaches lies in the arrival processes of workers and their contexts. To produce synthetic data, we generate workers and their contexts based on predefined distributions described below. In case of real data, similar to the approaches in [KS12, uHC14, ZC17], we use a data set from Gowalla [CML11]. Gowalla was a location-based social network where users shared their locations by checking in at "spots", i.e., certain places in their vicinity. We use the check-ins from the Gowalla data set to simulate the arrival process of workers and their contexts. The Gowalla data set used here consists of 6 442 892 check-ins of 107 092 distinct users over the period of February 2009 to October 2010. Each entry of the data set consists of the form (User ID, Check-in Time, Latitude, Longitude, Location ID). Similar to [ZC17], we first extract the check-ins in New York City, which leaves a subset of 138 954 check-ins of 7115 distinct users at 21 509 distinct locations. This resulting Gowalla-NY data set is used for the simulations below. Figures 5.3(a) and 5.3(b) show the distributions of the total number of check-ins per user and the number of distinct locations visited per user in the Gowalla-NY data set, respectively.

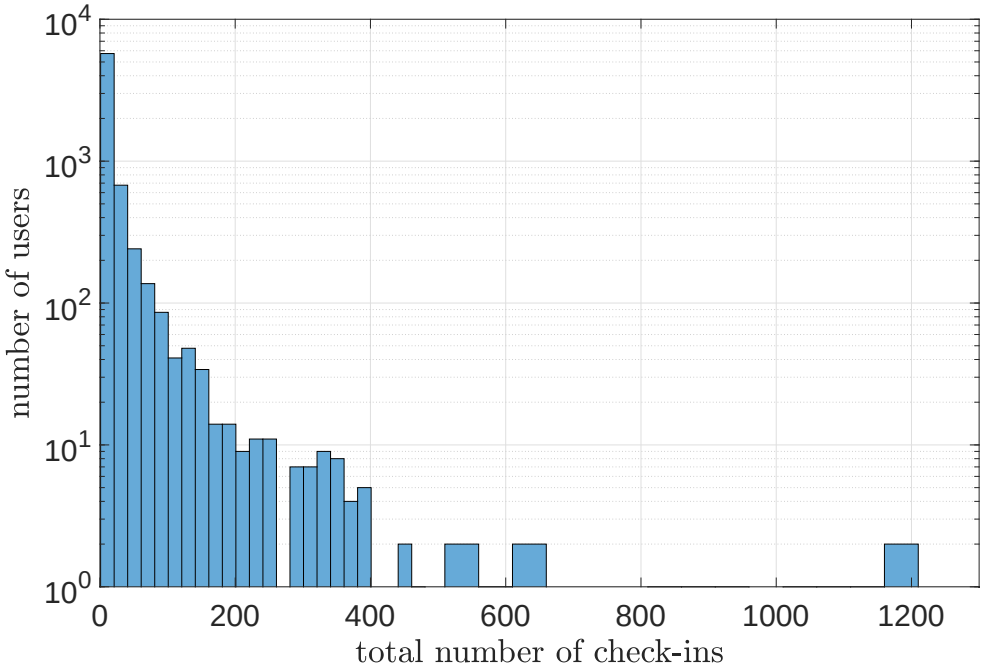
For both synthetic and real data, we simulate an MCSP, to which a set of $W = 100$ workers belongs. For synthetic data, 100 workers are created in the beginning. For real data, we randomly select 100 users from the Gowalla-NY data set, which represent the 100 workers of the MCS application. Then, we use this reduced Gowalla-NY data set containing the check-ins of 100 users.

5.7.1.2 Task Properties

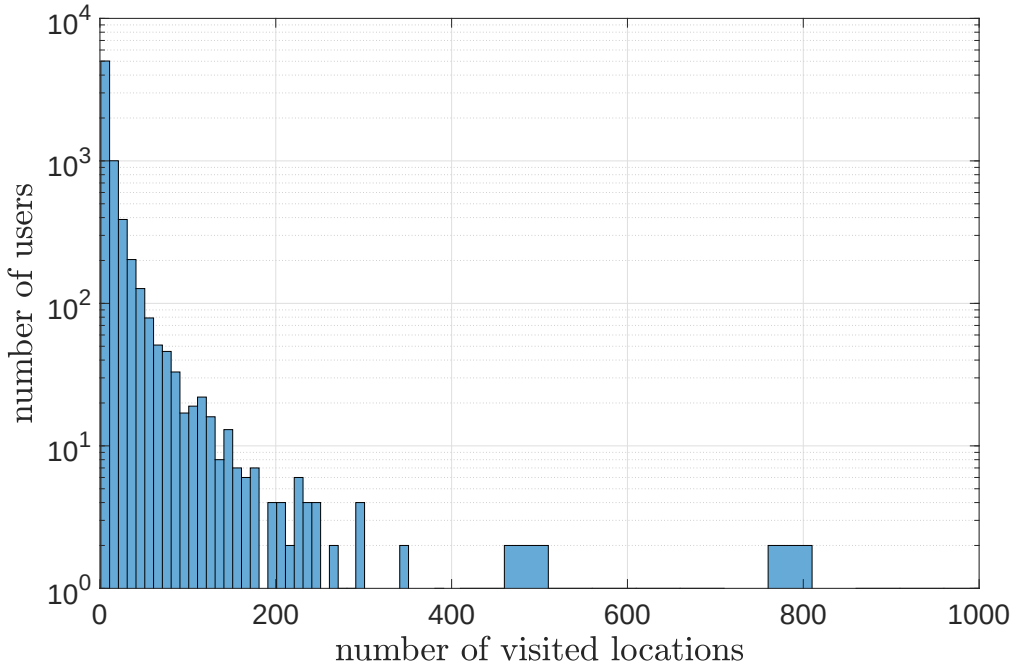
The task context is assumed to be uniformly distributed in $\mathcal{C} = [0, 1]$ (i.e., $C = 1$). Task owners have to pay a fixed price e_t of either 0.75 or 1 monetary units per requested worker that completes a task t, depending on whether the task context $\mathbf{c}_t$ lies in either $[0, 0.5]$ or $(0.5, 1]$. The quality of a completed task lies in the range $q_{\min} = 0$ and $q_{\max} = 5$. The task budget is sampled from a normal distribution with expected value 20 and standard deviation of 5, truncated between 1 and 100.

5.7.1.3 Worker Availability

For synthetic data, we let each worker be available with a probability of $\rho = 0.7$ (default value) for each arriving task. For the real data, we use a Binomial distribution with parameters $W = 100$ and $\rho = 0.7$ (default value) to sample the number of available workers W_t for an arriving task. In this way, the number of available workers in our



(a) Distribution of total number of check-ins of users in the data set.



(b) Distribution of number of distinct locations visited by users in the data set.

Figure 5.3. Statistics of Gowalla-NY data set.

experiments using the real and the synthetic data are distributed in the same way. For the real data, having sampled W_t, we randomly draw samples from the reduced Gowalla-NY data set (consisting of the check-ins of 100 users) until these samples contain W_t distinct users. These W_t sampled users correspond to the available workers at the arrival of task t. Hence, users with higher number of check-ins in the reduced Gowalla-NY data set translate to workers that are more often available for the MCSP.

5.7.1.4 Worker Context

The personal context space of worker i is set to $\mathcal{X}_i = [0,1]^2$ (i.e., $X_i = 2$). The first personal context dimension refers to the worker's battery state, which is sampled from a uniform distribution in $[0,1]$. The second personal context dimension refers to the worker's location, which is sampled differently in case of synthetic and real data. For synthetic data, the worker's location is sampled from 5 different (personal) locations, using a weighted discrete distribution with probabilities $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{12}, \frac{1}{24}, \frac{1}{24}\}$ to represent the fact that workers may spend more or less time using the MCS application in different places (e.g., at home more often than at work). For real data, we set the worker's location to be the check-in location of the respective user from the sample. If a user was sampled several times until we sampled W_t distinct users, we choose her/his first sampled check-in location.

5.7.1.5 Expected Worker Performance

We use two different models to generate expected worker performance.

Discrete Performance Model The joint personal and task context space $\mathcal{X}_i \times \mathcal{C}$ (of dimension $D_i = 3$) is split into a uniform grid. For synthetic data, the space is split into 5 identical parts along each of the 3 dimensions, i.e., $5 \cdot 5 \cdot 5 = 125$ subsets of $\mathcal{X}_i \times \mathcal{C}$ are created. For real data, along the dimensions of task context and battery state, the context space is split into 5 identical parts each, but along the dimension of location context, the context space of worker i is split into l_i identical parts, where l_i corresponds to the number of distinct locations visited by the corresponding user from the reduced Gowalla-NY data set. Hence, $5 \cdot 5 \cdot l_i$ subsets are created. Then, for both synthetic and real data, in each of the subsets, the expected performance of a worker is a priori sampled uniformly at random from $[0,5]$. Note that for the real data, since the expected performance differs per visited location, workers with higher number of visited locations have a higher number of different context-specific performances.

Hybrid Performance Model We assume a continuous dependency of the expected performance on two of the context dimensions. Let $x_i^{(1)}$ and $x_i^{(2)}$ be worker i's battery state and location, respectively, and let c be the task context. We assume that the expected performance θ_i of worker i is given by

$$\theta_i\left(c, x_i^{(1)}, x_i^{(2)}\right) = q_{\max} \cdot w_i\left(x_i^{(2)}\right) \cdot \bar{f}_{\mu_i,\sigma_i^2}(c) \cdot \sqrt{x_i^{(1)}}, \tag{5.15}$$

where $w_i\left(x_i^{(2)}\right)$ is a (discrete) location-specific weighting factor that is a priori sampled uniformly between $[0.5, 1]$ for each of worker i's (finitely many) locations. Moreover, $\bar{f}_{\mu_i,\sigma_i^2}$ is a truncated Gaussian probability density function with mean μ_i and standard deviation σ_i, which has been normalized such that its maximum value equals 1. For worker i, the mean μ_i is a priori sampled uniformly from $[0.1, 0.9]$ and the standard deviation is set to $\sigma_i = 0.1 \cdot \mu_i$. Hence, the expected performance is a continuous function of task context and battery state. The hybrid model has the following intuition. The expected performance of a worker is location-specific. Along the task context, the expected performance varies according to a worker-specific Gaussian distribution, i.e., each worker performs well at a specific type of tasks. Finally, the expected performance grows monotonically with the battery state, i.e., with more battery available, workers are more likely to perform well at tasks.

5.7.1.6 Instantaneous Worker Performance

For each occurring joint worker and task context, the instantaneous performance of a worker on an assigned task is sampled by adding noise uniformly sampled from $[-1, 1]$ to the expected performance in the given context. Note that the noise interval is truncated to a smaller interval if the expected performance lies close to either 0 or $q_{\max}$.

5.7.2 Reference Algorithms

The following algorithms are used for comparison.

- The *(centralized) Oracle* has perfect a priori knowledge about context-specific expected worker performances and knows the current contexts of all available workers. For each task t, the Oracle selects the best subset of workers according to (5.7).

- *LinUCB* is an algorithm for contextual MABs, cf. Section 2.3.3.4, that assumes that the expected performance of a worker is linear in its context [LCLS10], [CLRS11]. In the considered MCS problem, based on a linear reward function over contexts and previously observed context-specific worker performances, for each task, LinUCB chooses the m_t available workers with highest estimated upper confidence bounds on their expected performance. LinUCB has an input parameter λ_{LinUCB}, controlling the influence of the confidence bound. LinUCB is used in [uHC14] for task assignment in spatial CS.

- *AUER* is an extension of the well-known UCB1 algorithm, cf. Section 2.3.3.3, to the sleeping arm case [KNMS10]. It learns from previous observations of worker performances, but without taking into account context information. In the considered MCS problem, based on the history of previous observations of worker performances, AUER selects the m_t available workers with highest estimated upper confidence bounds on their expected performance. AUER has an input parameter λ_{AUER}, which controls the influence of the confidence bound.

- *ϵ-Greedy* is an algorithm for the stochastic MAB problem, cf. Section 2.3.3.3, that learns from the history of reward observations [ACBF02], but without taking context into account. In the considered MCS problem, ϵ-Greedy works as follows. ϵ-Greedy selects a random subset of available workers with a probability of $\epsilon \in (0, 1)$. With a probability of $(1 - \epsilon)$, ϵ-Greedy selects the m_t available workers with highest estimated performances. The estimated performance of a worker is computed as the sample mean of her/his previous performances.

- *Myopic* is a heuristic algorithm that learns only from the last interaction with each worker. For task 1, it selects a random subset of m_1 workers. For each of the following tasks, it checks which of the available workers have previously accepted a task. If more than m_t of the available workers have accepted a task when requested the last time, Myopic selects out of these workers the m_t workers with the highest performance in their last completed task. Otherwise, Myopic selects all of these workers and an additional subset of random workers so that in total m_t workers are selected.

- *Random* selects a random subset of m_t available workers for each task t.

Note that we have adapted the MAB algorithms LinUCB, AUER and ϵ-Greedy as these algorithms would have originally selected only one worker per task, while we required them to instead select m_t workers per task. Moreover, in the list above, we described the behavior of the five reference algorithms for the case $m_t < W_t$. In the case of $m_t \geq W_t$, we have adapted each of the five algorithms such that it selects all

Table 5.2. Choice of parameters for different algorithms.

Algorithm	**Parameter**	**Selected value**
HCL	λ_{HCL}	0.003
LinUCB	λ_{LinUCB}	1.5
AUER	λ_{AUER}	0.5
ϵ-Greedy	ϵ	0.01

available workers. Finally, while we used standard centralized implementations of the five reference algorithms, they could also be decoupled to a hierarchical setting like the one used by HCL.

5.7.3 Parameter Selection

The reference algorithms LinUCB, AUER and ϵ-Greedy each require an input parameter that affects the performance of the respective algorithm. Moreover, also in HCL, where we set $\alpha = 1$, choose $h_{T,i} = \lceil T^{\frac{1}{3+D_i}} \rceil$, $i \in \mathcal{W}$, as in Theorem 5.1, and set the control function to $K_i(t) = \lambda_{\text{HCL}} \cdot t^{\frac{2\alpha}{3\alpha+D_i}} \log(t)$, $t = 1, ..., T$, $i \in \mathcal{W}$, we have included the factor $\lambda_{\text{HCL}} \in (0, 1]$ into the control function as an additional input parameter to reduce the number of exploration phases. In order to find an appropriate input parameter for each of the four algorithms, we first generate 20 synthetic instances using the discrete performance model, where each instance consists of a sequence of $T = 10\,000$ task and worker arrivals sampled according to Section 5.7.1. Then, for each algorithm, we make a parameter sweeping, by running the algorithm for a range of different input parameters, each time averaging the results over the 20 instances, in order to find the parameter at which the algorithm on average performs best. Table 5.2 shows the parameters at which each of the algorithms on average performed best, respectively. These parameters are used in all of the following simulations.

5.7.4 Evaluation Metrics

Each algorithm is run over a sequence of tasks $t = 1, ..., T$ and its result is evaluated using the following metrics.

- We compute the *cumulative worker performance at* T achieved by an algorithm, which is the cumulative sum of performances by all selected workers up to (and

including) task T. Formally, if the set of selected workers of an algorithm A for task t is $\{s_{t,j}^A\}_{j=1,\dots,\min\{m_t,W_t\}}$ and $p_{s_{t,j}^A}(t)$ is the observed performance of worker $s_{t,j}^A$, the cumulative worker performance at T achieved by algorithm A is

$$\Gamma_T(A) := \sum_{t=1}^{T} \sum_{j=1}^{\min\{m_t,W_t\}} p_{s_{t,j}^A}(t). \tag{5.16}$$

- As a function of the arriving tasks, we compute the *average worker performance up to t* achieved by an algorithm, which is the average performance of all selected workers up to task t. Formally, it is defined by

$$\frac{1}{\sum_{\tilde{t}=1}^{t} \min\{m_{\tilde{t}}, W_{\tilde{t}}\}} \sum_{\tilde{t}=1}^{t} \sum_{j=1}^{\min\{m_{\tilde{t}},W_{\tilde{t}}\}} p_{s_{\tilde{t},j}^A}(\tilde{t}). \tag{5.17}$$

5.7.5 Results

5.7.5.1 Results under the Discrete Performance Model

First, we generate 100 synthetic and 100 real instances, in both cases using an availability probability of $\rho = 0.7$ and the discrete performance model. Each instance consists of a sequence of $T = 10\,000$ task and worker arrivals sampled according to Section 5.7.1. Then, we run the algorithms on these instances and average the results.

For both synthetic and real data, Table 5.3 compares the cumulative worker performance at T of an algorithm A with the one of HCL, by displaying $\Gamma_T(A)/\Gamma_T(\text{HCL})$. As expected, Random gives a lower bound on the achievable cumulative performance. Moreover, the results of LinUCB, AUER, ϵ-Greedy and Myopic lie close to the result of Random. This shows that algorithms which either do not take context into account (i.e., AUER, ϵ-Greedy and Myopic) or have a linearity assumption between context and performance (i.e., LinUCB), cannot cope with the non-linear dependency of expected worker performance on context. In contrast, HCL clearly outperforms LinUCB, AUER, ϵ-Greedy and Myopic, even though HCL observes worker performance only when requesting a worker for exploration purposes, while the other algorithms have access to worker performance whenever a worker is requested. This is due to the fact that HCL smartly exploits context. Moreover, HCL reaches a result close to the Oracle, which is an upper bound to the other algorithms due to its a priori knowledge. Comparing synthetic and real data, HCL has a better performance on the synthetic data, but it

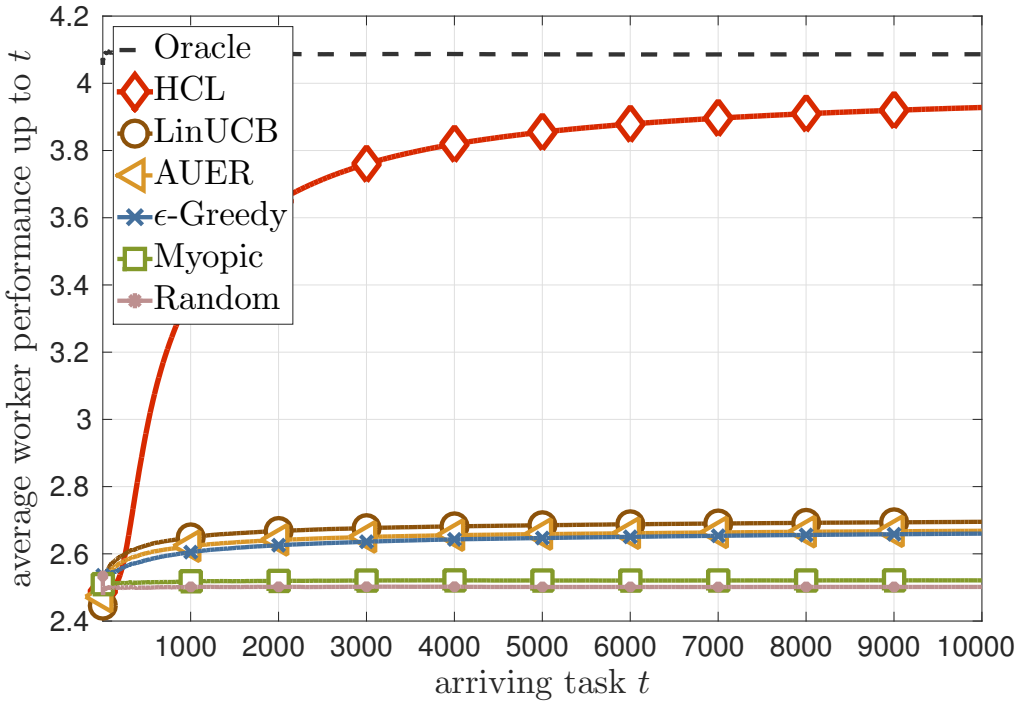
Table 5.3. Comparison of cumulative worker performance at T for $\rho = 0.7$ under the discrete performance model. For an algorithm A, the table shows $\Gamma_T(A)/\Gamma_T(\text{HCL})$.

Algorithm	**Synthetic data**	**Real data**
Oracle	1.04	1.20
HCL	1.00	1.00
LinUCB	0.69	0.78
AUER	0.68	0.77
ϵ-Greedy	0.68	0.76
Myopic	0.64	0.74
Random	0.64	0.73

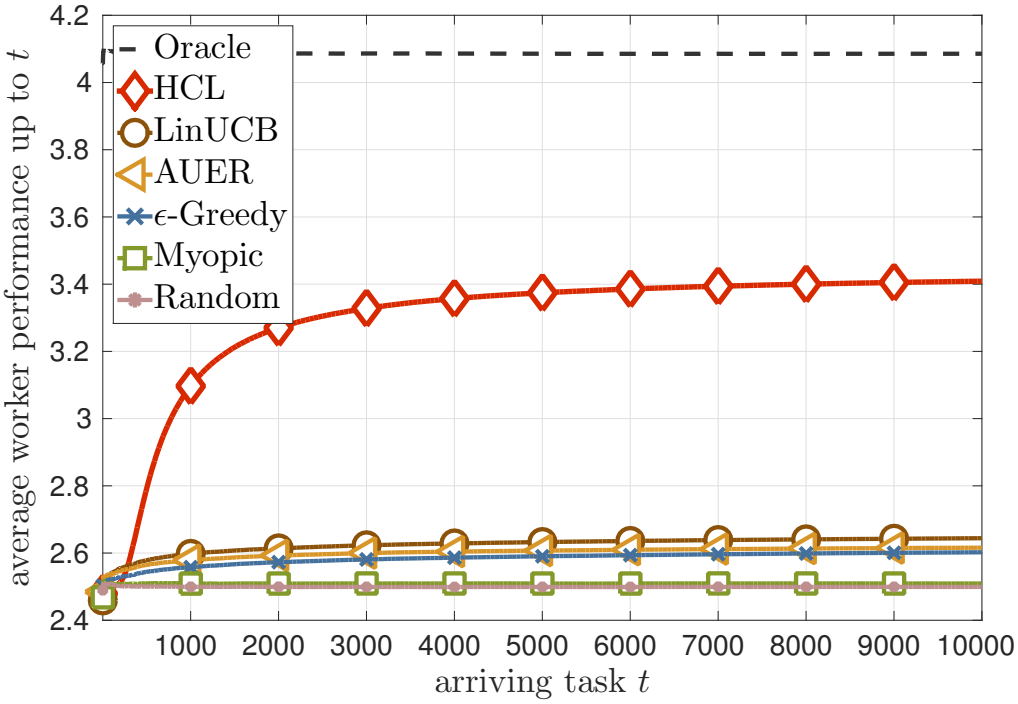
still reaches a good result on the real data, even though using real data, each worker has her/his own diversity in context arrival and hence in expected performance (since users in the Gowalla-NY data set have different numbers of visited check-in locations), i.e., for some workers, the context partition used by HCL may be more coarse than given by the worker's real context arrival process, while for others, it may be more fine granular.

Figures 5.4(a) and 5.4(b) show the average worker performance up to task t as a function of the sequentially arriving tasks $t = 1, ..., T$ in case of synthetic and real data, respectively. We see that over the sequence of tasks, the average worker performance achieved by Random and Oracle stay nearly constant at around 2.5 and 4.1, respectively, for both synthetic and real data. LinUCB, AUER, ϵ-Greedy and Myopic increase the average worker performance slightly, starting between 2.4 and 2.5 at $t = 1$ and ending with average performance of between 2.5 and 2.7 at $t = T$. On the contrary, HCL is able to increase the average worker performance from 2.5 at $t = 1$ up to 3.9 at $t = T$ for the synthetic data, and from 2.5 at $t = 1$ up to 3.4 at $t = T$ for the real data. Hence, HCL learns context-specific worker performances and selects better workers over time.

Finally, we evaluate the impact of worker availability by varying the parameter ρ. For each value of ρ, we average the results over 100 synthetic instances and over 100 real instances for $T = 10\,000$, respectively. Figures 5.5(a) and 5.5(b) show the cumulative worker performance at T achieved by the algorithms for different ρ in case of synthetic and real data, respectively. For small $\rho = 0.1$, all algorithms yield approximately the same performance. This is as expected since given our modeling of task budget, for small ρ, the number of available workers is often smaller than the required number of workers. Since each of the algorithms enters a select-all-workers phase in this case, each algorithm performs optimally. For increasing worker availability ρ, the cumulative performance at T achieved by each of the algorithms increases. However, the gap between

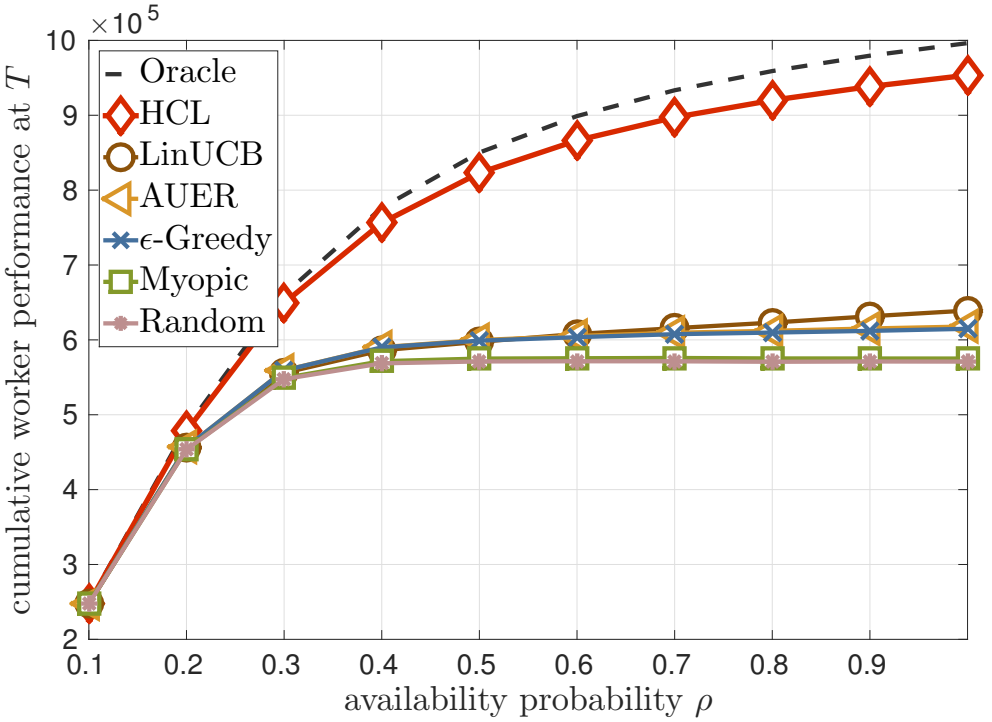


(a) Experiments with synthetic data.

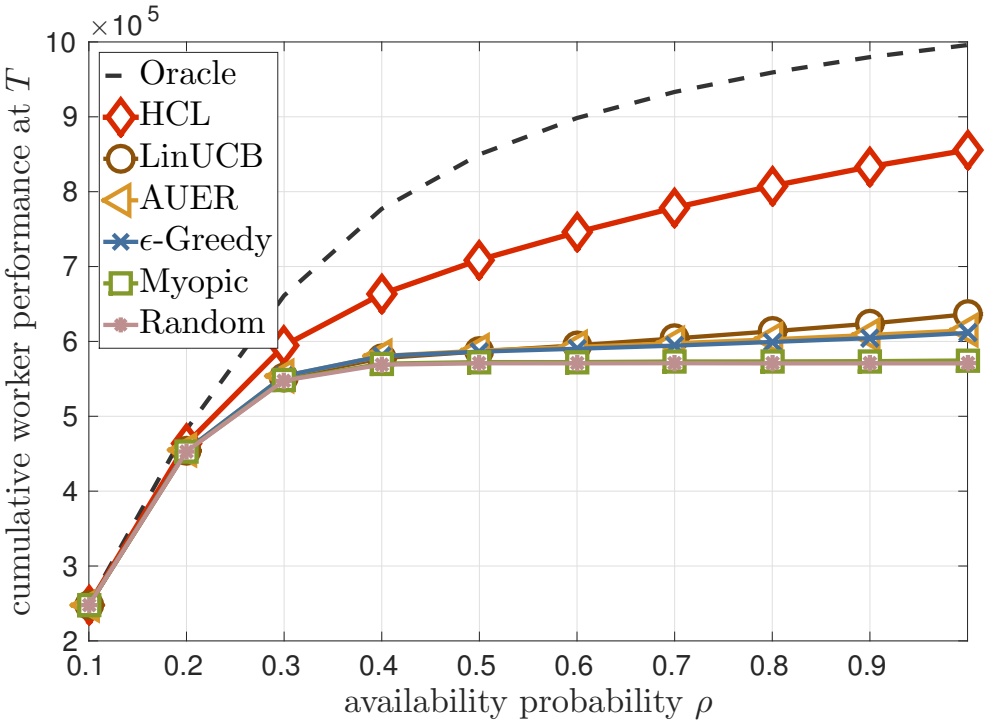


(b) Experiments with real data.

Figure 5.4. Average worker performance up to task t for sequence $t = 1, ..., T$ for $\rho = 0.7$ under the discrete performance model.



(a) Experiments with synthetic data.



(b) Experiments with real data.

Figure 5.5. Impact of worker availability on cumulative worker performance at T for $T = 10\,000$ tasks under the discrete performance model.

Oracle and HCL on the one hand, and the remaining algorithms on the other hand, is increasing for increasing ρ. For example, for the synthetic data, at $\rho \in \{0.3, 0.7, 1\}$, the cumulative performance achieved by HCL corresponds to $\{1.16, 1.46, 1.49\}$ times the one achieved by the respective next best algorithm {AUER, LinUCB, LinUCB}. Moreover, for the real data, at $\rho \in \{0.3, 0.7, 1\}$, the cumulative performance achieved by HCL corresponds to $\{1.07, 1.29, 1.34\}$ times the one achieved by the respective next best algorithm {ϵ-Greedy, LinUCB, LinUCB}. Hence, the more workers are available, the more severe is the effect of not selecting the best workers and only HCL is able to cope with the more difficult worker selection.

5.7.5.2 Results under the Hybrid Performance Model

Next, we evaluate the different algorithms under the hybrid performance model. Note that worker performance is differently distributed in the hybrid than in the discrete performance model, so that the absolute values in the results presented next are not comparable to those in Section 5.7.5.1.

First, we run the algorithms on 100 real instances for $T = 10\,000$ and $\rho = 0.7$ using the hybrid performance model. Figure 5.6 shows the average worker performance up to task t as a function of the sequentially arriving tasks $t = 1, ..., T$. The average worker performance achieved by Random and Oracle stay nearly constant at around 0.29 and 0.88 over the sequence of tasks. AUER, ϵ-Greedy and Myopic increase the average worker performance only slightly, from between 0.28 and 0.31 at $t = 1$ to between 0.36 and 0.42 at $t = T$. LinUCB has a larger increase from 0.37 at $t = 1$ to 0.55 at $t = T$. Compared to the discrete performance model, LinUCB performs better here due to the monotonic dependency of expected performance on battery state. Still, HCL has the largest increase from 0.31 at $t = 1$ up to 0.73 at $t = T$.

Finally, we evaluate the impact of worker availability ρ. For each value of ρ, we average the results over 100 real instances for $T = 10\,000$. Figure 5.7 shows the cumulative worker performance at T achieved by the algorithms for different ρ. Again, for higher ρ, the algorithms achieve higher cumulative performances at T. While LinUCB performs better compared to the results under the discrete performance model, still, the gap in cumulative performance between HCL and LinUCB is increasing for increasing ρ. For example, at $\rho \in \{0.3, 0.7, 1\}$, the cumulative performance achieved by HCL corresponds to $\{1.05, 1.32, 1.40\}$ times the one achieved by LinUCB.

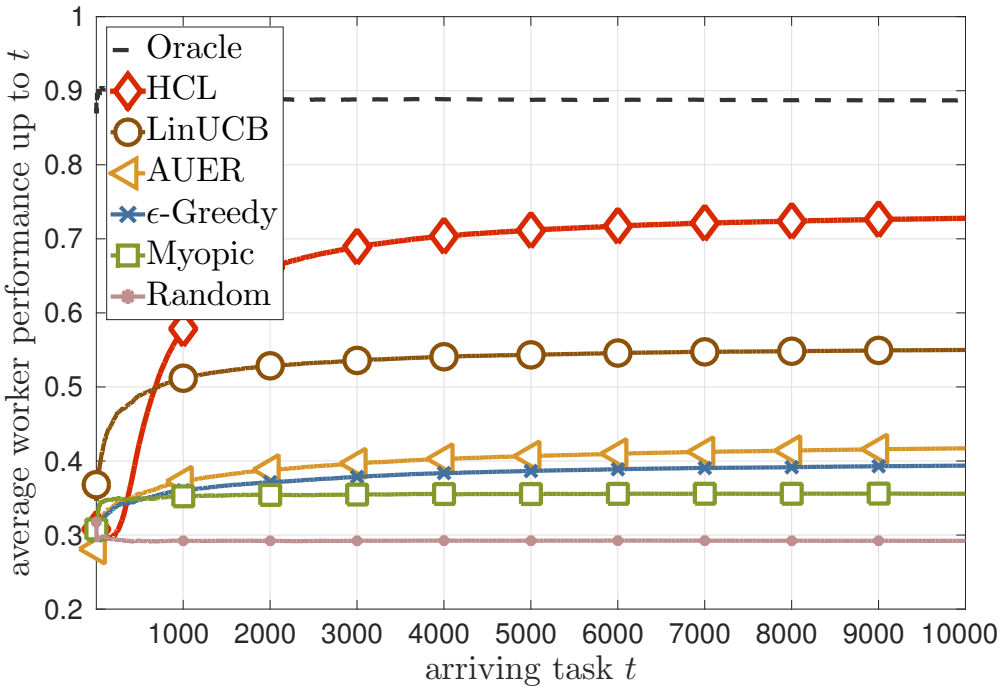


Figure 5.6. Average worker performance up to task t for sequence $t = 1, ..., T$ for $\rho = 0.7$ under the hybrid performance model using real data.

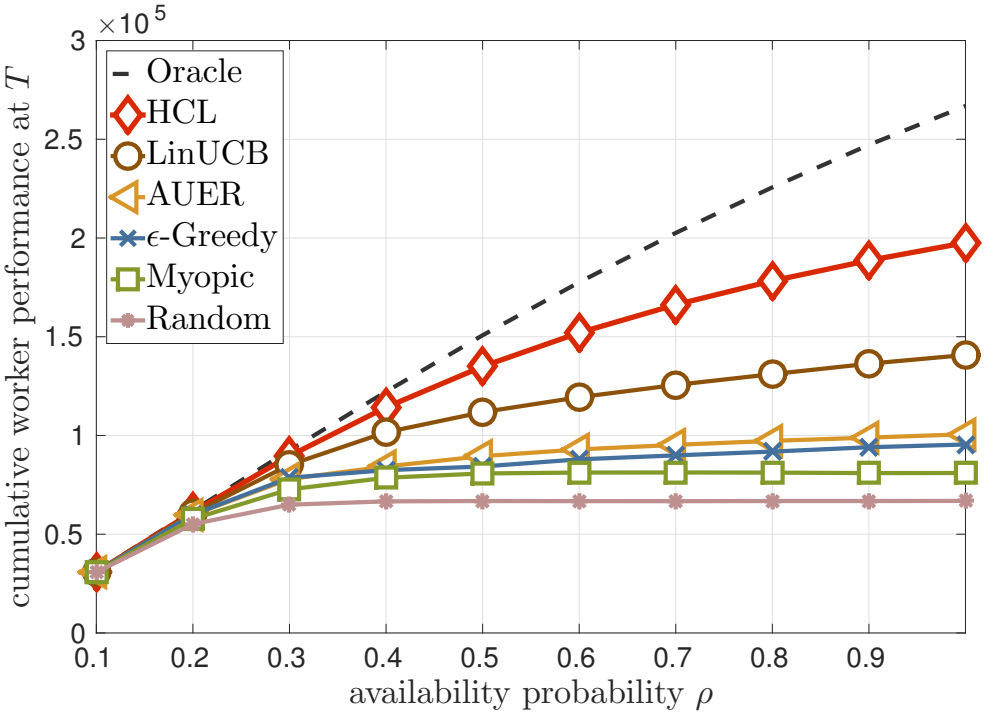


Figure 5.7. Impact of worker availability on cumulative worker performance at T for $T = 10\,000$ tasks under the hybrid performance model using real data.

5.8 Conclusions

In this chapter, we have investigated how to exploit user resources in wireless networks. Specifically, we have studied the problem of context-aware worker selection for maximizing the worker performance in an MCS application with non-spatial tasks under missing knowledge about each worker's individual performance. We have proposed a model for context-aware worker selection in MCS applications, which allows different task types to occur and which allows worker performance to be a possibly non-linear function of the task context and of the worker context. Then, taking a machine-learning-based approach, we have modeled the problem as a contextual MAB problem. Moreover, we have proposed a context-aware hierarchical online learning algorithm for worker selection in MCS applications based on a hierarchical architecture of decision making. In the proposed algorithm, decision making and information collection is split among different entities. On the one hand, LCs located in each of the workers' mobile devices learn their workers' performances online over time, by regularly observing the workers' personal contexts and their instantaneous performances. On the other hand, the centralized MCSP selects workers for tasks based on a regular information exchange with the LCs. This hierarchical coordination approach ensures that the most suitable workers are requested by the MCSP over time. Moreover, the learning in LCs ensures that personal worker context can be kept locally and does not need to be shared with the MCSP, but still workers are offered those tasks they are interested in the most.

The computational complexity of the algorithm has been shown to grow linearly with the dimension of the context space for the LCs and log-linearly with respect to the number of workers for the MCSP, respectively. Upper bounds on the local memory requirements of the proposed algorithm in the mobile devices as well as on the number of times the quality of each worker must be assessed have been derived. In addition, it has been shown that the more context dimensions the LCs are allowed to access, the lower are the communication requirements of the proposed hierarchical approach compared to an equivalent centralized approach. Besides, we have derived a sublinear upper bound on the regret, which analytically bounds the loss of the proposed algorithm with respect to an oracle that selects workers optimally under a priori knowledge about expected worker performance. The regret bound characterizes the learning speed and proves that the algorithm converges to the optimal worker selection strategy. Finally, simulations based on synthetic and real data have shown that, depending on the availability of workers, the proposed algorithm achieves an up to 49% higher cumulative worker performance than the best algorithm from the literature by smartly exploiting context information for worker selection.

Chapter 6

Conclusions

6.1 Summary

The contributions of this thesis can be summarized as follows. In this thesis, we understand wireless networks as *networks of distributed connected resources* – a recent *paradigm shift* that is mandatory in view of the expected increases in the amount of data traffic, the number of wirelessly connected devices and the requirements of emerging mobile and IoT applications, all of which will pose heavy burdens on future wireless networks. Following this new paradigm, new techniques are needed that jointly consider and leverage different types of resources available in wireless networks, namely, *communication, computation, caching, data collection* and *user resources*, in order to improve the system performance. In this thesis, it is shown that such new techniques require *context-aware decision making* in order to best exploit and allocate the different available resources. An overview of context-aware decision making is provided, by discussing *context awareness*, different types of *architectures of decision making* and different *designs of decision agents*. Finally, three candidate techniques for wireless networks are studied that jointly consider and leverage different types of resources, namely, *computation offloading in multi-hop wireless networks*, *caching at the edge of wireless networks* and *MCS*. For each technique, we identify a fundamental problem requiring context-aware decision making and we propose a novel framework for context-aware decision making that we use to solve the problem.

In Chapter 1, the need for the new paradigm of understanding wireless networks as networks of distributed connected resources is motivated. Moreover, the concept of context-aware decision making is introduced. Finally, three exemplary techniques are highlighted that jointly consider and leverage different types of resources of wireless networks. For each of the three techniques, a fundamental problem is identified and it is shown that context-aware decision making is required in order to best exploit the resources.

In Chapter 2, an overview of context-aware decision making in wireless networks is given. It is briefly outlined of which components a context-aware system model consists and the concept of context is introduced. Moreover, centralized, decentralized and hierarchical architectures of decision making are introduced and it is discussed for

which types of problems each of the architectures is suitable. Finally, different designs of decision agents and corresponding decision-making methods are discussed, with an emphasis on optimization and MAB frameworks, two specific types of approaches appearing in this thesis.

In Chapter 3, it is studied how to trade *computation resources* off against *communication resources* in wireless networks by considering computation offloading in multi-hop wireless networks. Using computation offloading, wirelessly connected devices may offload computation tasks to resource-rich servers for remote computation and thereby reduce their task completion times and their energy consumption. The effect of computation offloading on the energy consumption of an individual device depends not only on channel conditions and computing capabilities of the device, but also on task characteristics. Therefore, context information needs to be taken into account for deciding whether or not to offload a task. In this thesis, for the first time, we consider computation offloading in multi-hop networks, where network coverage may be extended and required transmission power reduced. Since communication resources of relay nodes need to be used and shared for task offloading, offloading decisions are non-trivially coupled in multi-hop networks. Therefore, in this chapter, the fundamental problem of context-aware computation offloading for energy minimization in multi-hop wireless networks is identified. First, a novel model for context-aware computation offloading in multi-hop wireless networks is proposed that takes into account channel conditions, computing capabilities of the devices, task characteristics, and battery constraints at relay nodes. Based on this model, using an *optimization-based approach*, the problem is formulated as a multi-dimensional knapsack problem, which takes into account the non-trivial coupling of offloading decisions. Then, using a *centralized architecture of decision making*, a new *context-aware greedy heuristic algorithm for computation offloading in multi-hop networks* is proposed. This algorithm enables a controller in the access point to take offloading decisions based on centrally collected information about network conditions and task context. The computational complexity of the proposed algorithm is analyzed and it is shown that the communication overhead of the proposed centralized architecture of decision making is small. Furthermore, numerical results demonstrate that the proposed algorithm on average reduces the network energy consumption by 13% compared to the case when no computation offloading is used. Moreover, the proposed algorithm yields near-optimal results in the considered offloading scenarios, with a maximal deviation of less than 6% from the global optimum.

In Chapter 4, it is investigated how to exploit *caching resources* in order to save *communication resources* in wireless networks by studying caching at the edge. Caching at the edge uses caching resources close to the mobile users to cache popular content in

a placement phase in order to locally serve user requests for this content in a delivery phase. In this way, the backhaul and cellular traffic may be alleviated and the latency for the user may be reduced. A crucial question is which content should be locally cached such that the number of cache hits is maximized. Caching the most popular content requires knowledge about the content popularity distribution, which is typically not available a priori. Moreover, local content popularity may vary according to the preferences of the mobile users connecting to the local cache over time. The users' preferences, in turn, may depend on their contexts. Finally, cache content placement needs to take into account the cache operator's specific objective, which may include the need for service differentiation. Hence, in this chapter, the fundamental problem of context-aware proactive caching for cache hit maximization at the edge of the wireless network under missing knowledge about content popularity is identified. First, a new model for context-aware proactive caching is introduced, allowing different content to be favored by different users and including that the content popularity depends on the user's context. Then, a *machine-learning-based approach* is pursued and the problem is modeled as a contextual MAB problem. Based on this model, a novel *online learning algorithm for context-aware proactive caching* is proposed using a *decentralized architecture of decision making*. This algorithm enables the controller of a local cache to learn context-specific content popularity online over time and to take service differentiation into account. The computational complexity and the memory and communication requirements of the proposed algorithm are analyzed and it is shown how the algorithm can be extended to practical requirements. Furthermore, a sublinear upper bound on the regret of the algorithm is derived, which characterizes the learning speed and proves that the proposed algorithm converges to the optimal cache content placement strategy. Finally, simulations based on real data show that, depending on the cache size, the proposed algorithm achieves up to 27% more cache hits than the best algorithm taken from the literature.

In Chapter 5, it is studied how to make use of *user resources* in wireless networks by considering MCS. Using MCS, task owners outsource their tasks via an intermediary MCSP to a set of workers, which allows different stakeholders to leverage human intelligence for task solving. Since different workers may have different interests and capabilities, not all of them may perform equally well on a given task. Hence, in order to maximize the worker performance on a given task under the task budget, the most suitable workers should be assigned to the task. Assigning the best workers to each task requires knowledge about the expected performance of each worker, which is typically not available a priori. Additionally, a worker's performance may depend not only on the specific task, but also on the worker's current context, and this dependency may be of non-linear nature. Furthermore, due to communication overhead and privacy con-

cerns of workers, it may be required to keep personal worker context locally instead of sharing it with the central MCSP, which makes it difficult for the MCSP to select the most suitable workers. Therefore, in this chapter, the fundamental problem of context-aware worker selection for maximizing the worker performance in an MCS application with non-spatial tasks under missing knowledge about each worker's individual performance is identified. First, a novel model for context-aware worker selection in MCS is proposed that allows different task types to occur and that allows worker performance to be a possibly non-linear function of the task context and of the worker context. Based on this model, a *machine-learning-based approach* is taken and the problem is modeled as a contextual MAB problem. Using a *hierarchical architecture of decision making*, a new *context-aware hierarchical online learning algorithm for worker selection in MCS* is proposed. In the proposed algorithm, decision making and information collection is split among different entities. While a set of LCs located in the workers' mobile devices learns the workers' context-specific performances online over time, the centralized MCSP assigns workers to tasks based on a regular information exchange with the LCs. This novel hierarchical coordination approach ensures that the most suitable workers are requested to complete the task by the MCSP over time, while personal worker context is kept locally in the LCs, thus preserving the workers' privacy and reducing communication overhead. The computational complexity of the proposed algorithm both for the LCs and the MCSP is analyzed. In addition, upper bounds on the local memory requirements of the proposed algorithm in the mobile devices as well as on the number of times the quality of each worker must be assessed are derived. Moreover, it is shown that the more access to worker context is granted to the LCs, the lower are the communication requirements of the proposed algorithm compared to an equivalent centralized approach. Furthermore, a sublinear upper bound on the regret is derived, which characterizes the learning speed and proves that the proposed algorithm converges to the optimal worker selection strategy. Finally, numerical results based on synthetic and real data show that, depending on the availability of workers, the proposed algorithm achieves an up to 49% higher cumulative worker performance than the best algorithm from the literature.

6.2 Outlook

We end this thesis with an outline of future research directions.

Computation Offloading In this thesis, we have proposed a context-aware greedy heuristic algorithm for computation offloading in multi-hop wireless networks, aiming

at minimizing the network energy consumption. The following extensions may be pursued in future. First, one may consider that nodes may process tasks for each other, i.e., considering a cooperative multi-hop network [FTH16]. In this case, for each node, it not only has to be decided whether or not to offload its task, but also to which other node to offload. Secondly, the approach may further be extended by changing the architecture of decision making to a decentralized one, where the nodes of the network take individual offloading decisions. In this case, a coordination mechanism is needed to ensure that the offloading decisions are valid and are aligned with the overall network goal. The author of this thesis has contributed to first steps in this direction in a follow-up paper [ASMK16]. Finally, we have assumed that the parameters appearing in the optimization problem are known. However, our approach could further be extended by no longer assuming that the ratio between the number of bits needed for transmission and the number of CPU cycles needed for computation are known a priori. Instead, this ratio would first have to be predicted as a function of the type of task and application, e.g., using machine-learning methods [KLLB13].

Caching at the Edge In this thesis, we have proposed an online learning algorithm for context-aware proactive caching, aiming at maximizing the number of cache hits in a local cache at the edge of the wireless network under missing knowledge about content popularity. Extensions in the following directions would be useful. First, we have considered that each content corresponds to one file in the library. However, in adaptive video streaming, videos are encoded into different representations that correspond to different qualities of the video [PIAT14]. How to assign different representations of different videos to multiple caches at the edge under a quality of experience metric is a challenging problem that has been studied in [LTZ+18] assuming a priori knowledge about video popularity. However, since video popularity is typically unknown a priori and has to be learned, adaptive video streaming should be studied in combination with context-aware proactive caching under missing knowledge about content popularity. Secondly, we have considered that the goal of the caching entity is to maximize the number of cache hits, but we did not consider the cost for placing content into the cache. Such a cost may occur due to the bandwidth consumption on the backhaul when the file to be cached is fetched from the distant file server. While a cache replacement cost has been considered for proactive caching under missing knowledge about content popularity in [BG14a, BG14c], these works did not take context into account for learning content popularity. Combining context-aware proactive caching with cache replacement costs is very challenging under arbitrary context arrivals since cache replacement costs require to keep the cache content static as often as possible, which is difficult if contexts arrive in an arbitrary manner. Therefore, cache replacement costs should be studied under suitable stochastic assumptions on the context arrival process

such that future changes in context can be estimated and taken into account for cache content replacement.

Mobile Crowdsourcing In this thesis, we have proposed a context-aware hierarchical online learning algorithm for worker selection in MCS applications with non-spatial tasks, aiming at maximizing the worker performance under missing knowledge about each worker's individual performance. This work could be extended as follows. First, in this thesis, we have considered that each worker is paid the same amount of compensation for completing a particular task. This could be extended by allowing workers to have a personal price as in [TTSRJ14] or even different personal prices for different task types. In this case, the problem becomes a combinatorial contextual MAB problem such that methods from combinatorial MABs [CWY13, QCZ14] need to be investigated to handle this problem. Secondly, we have considered that an LC uses available context information for learning worker performance. However, it may happen that not all of the available context dimensions are equally relevant for learning a worker's performance. Hence, in order to reduce the costs for monitoring and the required memory space for keeping monitored context information in the mobile device of its worker, an LC should additionally discover the most relevant context dimensions. For this purpose, how to learn relevance within contextual MABs needs to be studied [TvdS15b].

Context-Aware Decision Making in Wireless Networks In this thesis, we have seen that many problems in wireless networks require context awareness since the optimal decisions depend not only on the current network conditions, but also on other node-related, user-related or externally given conditions. The contextual MAB frameworks presented in this thesis are not restricted to the scenarios considered in this thesis, but can also be used to model and solve further problems of context-aware sequential decision making with limited feedback and missing a priori knowledge in wireless networks. The author of this thesis has already started to work in this direction and has contributed to publications that have adapted the contextual MAB framework from Chapter 4 to beam selection in 5G mmWave Vehicular Communications [AMS+18, SKA+18].

Distributed Connected Resources in Wireless Networks Overall, we have studied in this thesis how to exploit different available resources in order to improve the system performance of wireless networks. In detail, three techniques have been considered, each of which exploits one type of resources (i.e., computation, caching,

user resources) and considers its interplay with the communication resources, while relying on data collection resources to collect context information for decision making. Going, however, one step further, all the different types of resources, i.e., *communication*, *computation*, *caching*, *data collection* and *user resources*, should be considered *jointly* based on a joint modeling in order to understand their interrelationships and interdependencies. Moreover, a joint optimization of the allocation of all the different resources could possibly further improve the system performance. So far, approaches in this direction have considered the trade-offs between and the joint optimization of subsets of the above resource types, for instance, by jointly optimizing where to place and/or how to allocate computation, communication and caching resources [HYH+16, LCQ16, WZZ+17, CHH+18, WHY+18]. This approach may be extended as follows.

First, we have seen in this thesis that for optimally allocating computation, communication or caching resources, *context information* needs to be taken into account, which itself needs to be collected using *data collection resources*. Since the usage of data collection resources has a cost (e.g., energy or delay cost for sensing), one cannot simply continuously collect data in order to have all kinds of (possibly not needed) context available for decision making. Instead, finding and only relying on those sources which provide the relevant context is crucial [KLJ+10]. Moreover, computation resources are required in order to process collected context such that decision agents may actually use it, and caching resources are needed to store the collected context [MSS13]. Additionally, trade-offs between data collection resources (e.g., in terms of sensing) and communication resources exist [KASK19]. Overall, trade-offs between data collection resources and the other types of resources exist and we therefore argue that such trade-offs need to be included into the joint resource allocation of wireless networks. Since data collection has a cost, as in active sensing [YZvdS18], it needs to be further investigated, how much data collection resources need to be sacrificed in order to collect sufficient context information for taking well-informed decisions about the joint allocation of all the other types of resources. To sum up, data collection resources should be included into the joint optimization of resources in the same way as the more "traditional" communication, computation and caching resources.

Secondly, we have seen in this thesis how human intelligence may be leveraged within wireless networks. While users are already being perceived as service providers in the context of MCS [RZZS15], user resources should be more broadly understood as an essential type of resource available in wireless networks. For instance, users can actively take part in solving intelligence tasks (e.g., image annotation) that may be difficult for machines to solve or even intractable for machine computation [RZZS15]. In this way, human intelligence enhances the computing capabilities of the network.

As another example, users can actively take part in sensing tasks (e.g., environmental sensing) that might be solved very efficiently by a large set of users in the area of interest taking sensor readings with their mobile phones [HZL16]. In this way, human intelligence also enhances the data collection capabilities of the network. We argue, however, that rather than understanding user resources as part of the computing or data collection resources, user resources should be treated as a separate type of resource since human behavior introduces additional uncertainties (due to the users' preferences, skills and strategic behavior) that need to be taken into account.

Appendix

A.1 Proof of Proposition 3.1

In this appendix, we prove Proposition 3.1 from Section 3.5.1, proving that Problem (3.11) corresponds to a multi-dimensional knapsack problem.

Proof of Proposition 3.1. We turn the minimization Problem (3.11) into its equivalent maximization problem and we rewrite the energy constraints of Problem (3.11) to eliminate the index sets $\{n : r \in \mathcal{R}_n\}$. For that purpose, we define trivial coefficients $E^r_{\mathrm{T},n} := 0$ for all n, r with $r \notin \mathcal{R}_n$. Hence, Problem (3.11) is equivalent to the following problem:

$$\begin{aligned} -\max \ & \sum_{n=1}^{N} y_n (E_{\mathrm{C},n} - E_{\mathrm{T},n}) \\ \text{s.t.} \ & \sum_{n=1}^{N} y_n E^r_{\mathrm{T},n} \leq E_{\mathrm{prov},r} - E_{\mathrm{C},r} \text{ for } r = 1, ..., R \\ & y_n \in \{0, 1\} \text{ for } n = 1, ..., N. \end{aligned} \tag{A.1}$$

Setting the profits as $p_n := E_{\mathrm{C},n} - E_{\mathrm{T},n}$ for $n = 1, ..., N$, the weights as $w_{n,r} := E^r_{\mathrm{T},n}$ for $n = 1, ..., N$, $r = 1, ..., R$, and the capacity values as $c_r := E_{\mathrm{prov},r} - E_{\mathrm{C},r}$ for $r = 1, ..., R$, Problem (A.1) corresponds to a multi-dimensional knapsack problem as in (2.3) with R constraints. □

A.2 Proof of Proposition 3.3

In this appendix, we prove Proposition 3.3 from Section 3.5.5, thereby deriving the globally optimal actions of Problem (3.11) in the case of a line topology.

Proof of Proposition 3.3. First, assume that $E_{\mathrm{prov}} = E_{\mathrm{C}}$ holds. In this case, Rule 2 from Section 3.5.3 states that $y^*_n = 0$ for nodes $n = 2, ..., N$. Moreover, by Rule 1 from Section 3.5.3, $y^*_1 = 0$ holds if $E_{\mathrm{link}} \geq E_C$. Clearly, if $E_{\mathrm{link}} < E_C$, then $y^*_1 = 1$. Hence, it follows that

$$y^*_n = \begin{cases} 1, & \text{if } n < \frac{E_{\mathrm{C}}}{E_{\mathrm{link}}} \text{ and } n \leq \frac{E_{\mathrm{prov}} - E_{\mathrm{C}}}{E_{\mathrm{link}}} + 1 \\ 0, & \text{else.} \end{cases} \tag{A.2}$$

Now, assume without loss of generality that $E_{\text{prov}} > E_{\text{C}}$ holds. In a line topology, predecessors of node n are all nodes with smaller hop distance to the AP, i.e., nodes 1 to $n-1$. Hence, in the homogeneous setting of Proposition 3.3, the energy $E_{\text{T},n}$ for transmission of node n's task to the server in (3.6) reduces to

$$E_{\text{T},n} = E^n_{\text{T},n} + \sum_{r=1}^{n-1} E^r_{\text{T},n} = E_{\text{link}} + \sum_{r=1}^{n-1} E_{\text{link}} = n \cdot E_{\text{link}}. \tag{A.3}$$

Using Equation (A.3) and the fact that the successor nodes of a relay node r, $r = 1, ..., N-1$, are given by all nodes which have higher hop distance to the AP, i.e., nodes $r+1$ to N, Problem (3.11) becomes

$$\begin{aligned} \min \ & \sum_{n=1}^{N} y_n (n E_{\text{link}} - E_{\text{C}}) \\ \text{s.t.} \ & \sum_{n=r+1}^{N} y_n E_{\text{link}} \leq E_{\text{prov}} - E_{\text{C}} \text{ for } r = 1, ..., N-1 \\ & y_n \in \{0, 1\} \text{ for } n = 1, ..., N. \end{aligned} \tag{A.4}$$

As discussed in pre-processing Rule 1 in Section 3.5.3, each node whose optimal action it is to transmit its task to the server must have $E_{\text{T},n} < E_{\text{C}}$. By Equation (A.3), this is equivalent to $n \cdot E_{\text{link}} < E_{\text{C}}$ or $n < \frac{E_{\text{C}}}{E_{\text{link}}}$. The concept of dominance [KPP04] may now be used to prove which variables y_n are nonzero in the optimal solution, i.e., to find out for which node the optimal action is to use computation offloading. Translated to the computation offloading scenario, a node n dominates a node k if (i) node n provides at least as much energy savings when using computation offloading as node k and if (ii) node n needs at most as many energy resources from any relay node in the network as node k when using computation offloading. Both conditions are satisfied if the nodes' indices satisfy $n \leq k$ since (i) then the objective values satisfy $nE_{\text{link}} - E_{\text{C}} \leq kE_{\text{link}} - E_{\text{C}}$ and since (ii) nodes n and k need the same amount of energy from common relay nodes by homogeneity, but node n has lower hop distance to the AP than node k and thus needs energy of fewer relay nodes. Hence, a node n dominates all nodes k with $k \geq n$. Therefore, a dominance ordering of the nodes arises according to their hop distance to the AP. Hence, starting from node 1, which dominates all other nodes, one may set $y_n = 1$ for one node after the other, as long as this does not violate any of the energy constraints. Node 1 has the tightest energy constraint since among the relay nodes, which all have the same energy available, node 1 has the highest number of successor nodes. Suppose nodes 1 to $n-1$ were already chosen to offload their tasks, i.e., $y_k = 1$ for $k = 1, ..., n-1$. Then, enough energy is available for node n to transmit if

$$\sum_{k=2}^{n} 1 \cdot E_{\text{link}} \leq E_{\text{prov}} - E_{\text{C}}, \tag{A.5}$$

or, equivalently,

$$n - 1 \leq \frac{E_{\text{prov}} - E_{\text{C}}}{E_{\text{link}}}. \tag{A.6}$$

To sum up, the optimal action vector is hence given by

$$y_n^* = \begin{cases} 1, & \text{if } n < \frac{E_{\text{C}}}{E_{\text{link}}} \text{ and } n \leq \frac{E_{\text{prov}} - E_{\text{C}}}{E_{\text{link}}} + 1 \\ 0, & \text{else.} \end{cases} \tag{A.7}$$

This concludes the proof. □

A.3 Proof of Proposition 3.4

In this appendix, we prove Proposition 3.4 from Section 3.7.1, showing that the context-aware greedy heuristic algorithm in Algorithm 3.1 always selects the globally optimal actions in case of a star topology.

Proof of Proposition 3.4. During pre-processing, Algorithm 3.1 applies Rule 1 from Section 3.5.3 by setting $y_n = 0$ for any node n with $E_{\text{T},n} \geq E_{\text{C},n}$. After pre-processing, each remaining node n satisfies $E_{\text{T},n} < E_{\text{C},n}$. Algorithm 3.1 then sorts the remaining nodes according to the efficiency measure and, one after another, a remaining node n is added to the set of transmitting nodes by setting its variable to $y_n = 1$. Since no relay node exists in the star topology and hence, no energy constraints occur in the corresponding Problem (3.11), all remaining nodes are added to the set of transmitting nodes. Hence, Algorithm 3.1 sets $y_n = 1$ for any node n with $E_{\text{T},n} < E_{\text{C},n}$. To sum up, the output of Algorithm 3.1 is

$$y_n = \begin{cases} 1, & \text{if } E_{\text{T},n} < E_{\text{C},n} \\ 0, & \text{if } E_{\text{T},n} \geq E_{\text{C},n}, \end{cases} \tag{A.8}$$

which corresponds exactly to the optimal actions for a star topology according to (3.12) in Proposition 3.2. □

A.4 Proof of Proposition 3.5

In this appendix, we prove Proposition 3.5 from Section 3.7.1, showing that the context-aware greedy heuristic algorithm in Algorithm 3.1 always selects the globally optimal actions in case of a homogeneous line topology.

Proof of Proposition 3.5. First, assume that $E_{\text{prov}} = E_{\text{C}}$ holds. In this case, Rule 2 applies during pre-processing. In the homogeneous line setting of Proposition 3.5, Rule 2 reduces to the following: If $E_{\text{prov}} = E_{\text{C}}$, then $y_n = 0$ for nodes $n = 2, ..., N$. Hence, in this case, Algorithm 3.1 selects $y_n = 0$ for nodes $n = 2, ..., N$ and it selects $y_1 = 0$ if $E_{\text{link}} \geq E_C$ using pre-processing Rule 1 and $y_1 = 1$ otherwise. Hence, formally in this case, Algorithm 3.1 selects nodes according to

$$y_n = \begin{cases} 1, & \text{if } n < \frac{E_{\text{C}}}{E_{\text{link}}} \text{ and } n \leq \frac{E_{\text{prov}} - E_{\text{C}}}{E_{\text{link}}} + 1 \\ 0, & \text{else,} \end{cases} \tag{A.9}$$

which corresponds exactly to the optimal actions for a homogeneous line topology according to (3.16) in Proposition 3.3.

Now, assume without loss of generality that $E_{\text{prov}} > E_{\text{C}}$ holds. In a line topology, predecessors of node n are all nodes with smaller hop distance to the AP, i.e., nodes 1 to $n-1$. Hence, in the homogeneous setting of Proposition 3.5, the energy $E_{\text{T},n}$ for transmission of node n's task to the server in (3.6) reduces to

$$E_{\text{T},n} = E_{\text{T},n}^{n} + \sum_{r=1}^{n-1} E_{\text{T},n}^{r} = E_{\text{link}} + \sum_{r=1}^{n-1} E_{\text{link}} = n \cdot E_{\text{link}}. \tag{A.10}$$

During pre-processing, Algorithm 3.1 applies Rule 1 from Section 3.5.3 by setting $y_n = 0$ for any node n with $E_{\text{T},n} \geq E_{\text{C},n}$. Using (A.10), this is equivalent to $n \geq \frac{E_{\text{C},n}}{E_{\text{link}}}$. The remaining nodes, for which $n < \frac{E_{\text{C}}}{E_{\text{link}}}$ must hold, are sorted according to their efficiency measure. Using (A.10) and using the fact that $\mathcal{R}_n = \{1, ..., n-1\}$ in a line topology, the efficiency measure for a node n is given by

$$\begin{aligned} \text{eff}_n &= \frac{E_{\text{C},n} - E_{\text{T},n}}{\sum_{r \in \mathcal{R}_n} \frac{E_{\text{T},n}^{r}}{E_{\text{prov},r} - E_{\text{C},r}}} \\ &= \frac{E_{\text{C}} - n \cdot E_{\text{link}}}{\sum_{r=1}^{n-1} \frac{E_{\text{link}}}{E_{\text{prov}} - E_{\text{C}}}} \\ &= \frac{E_{\text{C}} - n \cdot E_{\text{link}}}{(n-1) \cdot \frac{E_{\text{link}}}{E_{\text{prov}} - E_{\text{C}}}} \end{aligned} \tag{A.11}$$

For any $n < j$, we have $E_{\text{C}} - n \cdot E_{\text{link}} > E_{\text{C}} - j \cdot E_{\text{link}}$ and $(n-1) \cdot \frac{E_{\text{link}}}{E_{\text{prov}} - E_{\text{C}}} < (j-1) \cdot \frac{E_{\text{link}}}{E_{\text{prov}} - E_{\text{C}}}$. Therefore, the efficiency measures eff_n and eff_j of any two nodes n and j with $n < j$ satisfy $\text{eff}_n > \text{eff}_j$. Hence, Algorithm 3.1 sorts the nodes with $n < \frac{E_{\text{C}}}{E_{\text{link}}}$ according to their hop distance to the AP, where the node with the smallest hop distance has the highest efficiency. Then, starting from the node with lowest hop distance, Algorithm 3.1 sets $y_n = 1$ for one node after the other, as long as this does not violate any of the energy constraints. Node 1 has the tightest energy constraint

since among the relay nodes, which all have the same energy available, node 1 has the highest number of successor nodes. Suppose nodes 1 to $n-1$ were already chosen to offload their tasks, i.e., $y_k = 1$ for $k = 1, ..., n-1$. Then, enough energy is available for node n to transmit if

$$\sum_{k=2}^{n} 1 \cdot E_{\text{link}} \leq E_{\text{prov}} - E_{\text{C}}, \tag{A.12}$$

or, equivalently,

$$n \leq \frac{E_{\text{prov}} - E_{\text{C}}}{E_{\text{link}}} + 1. \tag{A.13}$$

To sum up, Algorithm 3.1 selects the actions as

$$y_n = \begin{cases} 1, & \text{if } n < \frac{E_{\text{C}}}{E_{\text{link}}} \text{ and } n \leq \frac{E_{\text{prov}} - E_{\text{C}}}{E_{\text{link}}} + 1 \\ 0, & \text{else,} \end{cases} \tag{A.14}$$

which corresponds exactly to the optimal actions for a homogeneous line topology according to (3.16) in Proposition 3.3. □

A.5 Proof of Theorem 4.1

In this appendix, we prove Theorem 4.1 from Section 4.6.1, showing that the regret of CAC is sublinear in the time horizon T, which guarantees that CAC converges to the oracle solution for $T \to \infty$. First, two lemmas are proved and then the results of the two lemmas are combined to conclude the proof.

Given a sequence of T time slots with arbitrary user arrivals, let $\tilde{\tau}_T \subseteq \{1, ..., T\}$ be the set of time slots in which CAC enters an exploitation phase, and let $\tilde{\tau}_T^c = \{1, ..., T\} \setminus \tilde{\tau}_T$ be the set of time slots in which CAC enters an exploration phase. The sets $\tilde{\tau}_T$ and $\tilde{\tau}_T^c$ are random sets that depend on the cache selections made by CAC and the randomness of the observed demands. Let $R_{\text{or}}(T)$ and $R_{\text{oi}}(T)$ represent the regret due to exploration phases and due to exploitation phases, respectively. Using the expressions above, the regret $R(T)$ in (4.7) can be decomposed as follows:

$$R(T) = \mathbb{E}\left[R_{\text{or}}(T) + R_{\text{oi}}(T)\right], \tag{A.15}$$

where

$$R_{\text{or}}(T) := \sum_{t \in \tilde{\tau}_T^c} \sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \left(w_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)} \mu_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)}(\mathbf{x}_{t,i}) - w_{c_{t,j}} \mu_{c_{t,j}}(\mathbf{x}_{t,i}) \right) \tag{A.16}$$

$$R_{\text{oi}}(T) := \sum_{t \in \tilde{\tau}_T} \sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \left(w_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)} \mu_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)}(\mathbf{x}_{t,i}) - w_{c_{t,j}} \mu_{c_{t,j}}(\mathbf{x}_{t,i}) \right). \tag{A.17}$$

The regret is computed by considering the loss due to caching a subset of files $\{c_{t,j}\}_{j=1,...,m}$ instead of the top-m files for pair $(\mathcal{X}_t, \mathcal{G}_t)$ for each time slot. The loss is given by substracting the sum of weighted expected demands of the top m-files from the sum of weighted expected demands of the selected files.

Next, we will bound the expected values of the two summands above separately.

First, a bound for $\mathbb{E}[R_{\text{or}}(T)]$ is given.

Lemma A.1 (Bound for $\mathbb{E}[R_{\text{or}}(T)]$)**.** *Let $K(t) = t^z \log(t)$, $t = 1, ..., T$, and $h_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{D}$. If CAC is run with these parameters, the regret $\mathbb{E}[R_{\text{or}}(T)]$ is bounded by*

$$\mathbb{E}[R_{\text{or}}(T)] \leq mU_{\max} v_{\max} w_{\max} R_{\max} 2^D |\mathcal{F}| \cdot (\log(T) T^{z+\gamma D} + T^{\gamma D}). \tag{A.18}$$

Proof of Lemma A.1. Let $t \in \tilde{\tau}_T^c$ be a time slot for which CAC enters an exploration phase. Since the expected demand $\mu_f(\mathbf{x})$ for any $f \in \mathcal{F}$, $\mathbf{x} \in [0,1]^D$ is bounded in $[0, R_{\max}]$ and the service weights are bounded by $v_{\max}$ for the different service groups and by $w_{\max}$ for the prioritization weights, it follows that

$$\begin{aligned} R_{\text{or}}(T) &= \sum_{t \in \tilde{\tau}_T^c} \sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \Big((w_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)} \mu_{f_j^*(\mathcal{X}_t, \mathcal{G}_t)}(\mathbf{x}_{t,i}) - w_{c_{t,j}} \mu_{c_{t,j}}(\mathbf{x}_{t,i}) \Big) \\ &\leq \sum_{t \in \tilde{\tau}_T^c} mU_{\max} v_{\max} w_{\max} R_{\max}. \end{aligned} \tag{A.19}$$

Hence, the regret can be bounded by

$$\begin{aligned} \mathbb{E}[R_{\text{or}}(T)] &\leq \mathbb{E}\left[\sum_{t \in \tilde{\tau}_T^c} mU_{\max} v_{\max} w_{\max} R_{\max} \right] \\ &= mU_{\max} v_{\max} w_{\max} R_{\max} \, \mathbb{E}\left[\sum_{t \in \tilde{\tau}_T^c} 1 \right]. \end{aligned} \tag{A.20}$$

For $t \in \tilde{\tau}_T^c$, the set of under-explored files $\mathcal{F}_t^{\text{ue}}$ is non-empty, i.e., there exist a user $i \in \{1, ..., U_t\}$ with corresponding set $p_{t,i}$, and a file $f \in \mathcal{F}$, for which $N_{f,p_{t,i}}(t) \leq K(t) = t^z \log(t)$ holds. By definition of $\mathcal{F}_t^{\text{ue}}$, up to time slot T, there can be at most $\lceil T^z \log(T) \rceil$ exploration phases in which file f is selected due to a context from set $p_{t,i}$. Since there are $(h_T)^D$ hypercubes in the partition, there can be at most $(h_T)^D \lceil T^z \log(T) \rceil$

exploration phases in which file f is selected due to its under-exploration. Hence, the number of exploration phases is upper-bounded as follows:

$$\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T^c} 1\right] \leq \sum_{f\in\mathcal{F}} (h_T)^D \lceil T^z \log(T)\rceil. \tag{A.21}$$

Note that this upper bound is rather loose because several files may be explored simultaneously, in which case they do not induce separate exploration phases. Further, we conclude

$$\mathbb{E}[R_{\text{or}}(T)] \leq mU_{\max}v_{\max}w_{\max}R_{\max} \sum_{f\in\mathcal{F}} (h_T)^D \lceil T^z \log(T)\rceil. \tag{A.22}$$

Using $(h_T)^D = \lceil T^\gamma \rceil^D \leq (2T^\gamma)^D = 2^D T^{\gamma D}$, we have

$$\mathbb{E}[R_{\text{or}}(T)] \leq mU_{\max}v_{\max}w_{\max}R_{\max}2^D|\mathcal{F}| \cdot (\log(T)T^{z+\gamma D} + T^{\gamma D}). \tag{A.23}$$

□

Next, we give a bound for $\mathbb{E}\left[R_{\text{oi}}(T)\right]$.

Lemma A.2 (Bound for $\mathbb{E}\left[R_{\text{oi}}(T)\right]$)**.** *Let $K(t) = t^z \log(t)$, $t = 1, ..., T$, and $h_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{D}$. If CAC is run with these parameters and Assumption 4.1 from Section 4.6.1 holds true, the regret $\mathbb{E}\left[R_{\text{oi}}(T)\right]$ is bounded by*

$$\begin{aligned}\mathbb{E}[R_{\text{oi}}(T)] \leq\ & 2mU_{\max}v_{\max}w_{\max}R_{\max}\frac{T^{1-\frac{z}{2}}}{1-\frac{z}{2}} + 2mU_{\max}v_{\max}w_{\max}LD^{\frac{\alpha}{2}}T^{1-\alpha\gamma} \\ & + mU_{\max}^2 v_{\max}w_{\max}R_{\max}|\mathcal{F}|\frac{\pi^2}{3}.\end{aligned} \tag{A.24}$$

Proof of Lemma A.2. Let $t \in \tilde{\tau}_T$, i.e., CAC enters an exploitation phase. Since the set of under-explored files is empty in exploitation phases (i.e., $\mathcal{F}_t^{\text{ue}} = \emptyset$), $N_{f,p_{t,i}}(t) > K(t) = t^z \log(t)$ holds for all $f \in \mathcal{F}$ and all $i = 1, ..., U_t$.

Now, let $V(t)$ be the event that in time slot t, the estimated demand $\hat{\mu}_{f,p_{t,i}}(t)$ of each file $f \in \mathcal{F}$ in each of the current hypercubes $p_{t,i}$, $i = 1, ..., U_t$, is "close" to its true expected value $\mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]$, i.e.,

$$V(t) = \{|\hat{\mu}_{f,p_{t,i}}(t) - \mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]| < H(t) \text{ for all } f \in \mathcal{F}, i = 1, ..., U_t\} \tag{A.25}$$

for an arbitrary $H(t) > 0$.

Next, we distinguish between exploitation phases in which $V(t)$ or its complementary event, denoted by $V^c(t)$, hold. Let $I_{\{\cdot\}}$ denote the indicator function. Then, we can write

$$\begin{aligned}
&R_{\text{oi}}(T)\\
&= \sum_{t\in\tilde{\tau}_T}\Bigg(I_{\{V(t)\}}\Bigg(\sum_{j=1}^{m}\sum_{i=1}^{U_t} v_{g_{t,i}}\Big(w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}\mu_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}(\mathbf{x}_{t,i}) - w_{c_{t,j}}\mu_{c_{t,j}}(\mathbf{x}_{t,i})\Big)\Bigg)\Bigg)\\
&\quad+\sum_{t\in\tilde{\tau}_T}\Bigg(I_{\{V^c(t)\}}\Bigg(\sum_{j=1}^{m}\sum_{i=1}^{U_t} v_{g_{t,i}}\Big(w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}\mu_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}(\mathbf{x}_{t,i}) - w_{c_{t,j}}\mu_{c_{t,j}}(\mathbf{x}_{t,i})\Big)\Bigg)\Bigg). \qquad \text{(A.26)}
\end{aligned}$$

Using that the expected demand $\mu_f(\mathbf{x})$ for any $f\in\mathcal{F}$, $\mathbf{x}\in[0,1]^D$ is bounded in $[0,R_{\max}]$ and the service weights are bounded by $v_{\max}$ for the different service groups and by $w_{\max}$ for the prioritization weights, this term can further be bounded as

$$\begin{aligned}
&R_{\text{oi}}(T)\\
&\leq\sum_{t\in\tilde{\tau}_T}\Bigg(I_{\{V(t)\}}\cdot\Bigg(\sum_{j=1}^{m}\sum_{i=1}^{U_t} v_{g_{t,i}}\Big(w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}\mu_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}(\mathbf{x}_{t,i}) - w_{c_{t,j}}\mu_{c_{t,j}}(\mathbf{x}_{t,i})\Big)\Bigg)\Bigg)\\
&\quad+\sum_{t\in\tilde{\tau}_T} mU_{\max}v_{\max}w_{\max}R_{\max}I_{\{V^c(t)\}}. \qquad \text{(A.27)}
\end{aligned}$$

First, we bound the first term in (A.27). We start by noting that in an exploitation phase $t\in\tilde{\tau}_T$, since CAC selected files $\{c_{t,j}\}_{j=1,\dots,m}$ instead of $\{f_j^*(\mathcal{X}_t,\mathcal{G}_t)\}_{j=1,\dots,m}$, we have

$$\sum_{j=1}^{m}\sum_{i=1}^{U_t} v_{g_{t,i}}w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}\hat{\mu}_{f_j^*(\mathcal{X}_t,\mathcal{G}_t),p_{t,i}}(t) \leq \sum_{j=1}^{m}\sum_{i=1}^{U_t} v_{g_{t,i}}w_{c_{t,j}}\hat{\mu}_{c_{t,j},p_{t,i}}(t). \qquad \text{(A.28)}$$

We also know that when $V(t)$ holds, we have

$$\{|\hat{\mu}_{f,p_{t,i}}(t) - \mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]| < H(t) \text{ for all } f\in\mathcal{F}, i=1,\dots,U_t\} \qquad \text{(A.29)}$$

almost surely. Finally, note that by the Hölder continuity from Assumption 4.1, since $\mathbf{x}_{t,i}\in p_{t,i}$ and for calculating $\hat{\mu}_{f,p_{t,i}}(t)$, only contexts from hypercube $p_{t,i}$ are

used, for each $f \in \mathcal{F}$, it follows that

$$
\begin{aligned}
&|\mu_f(\mathbf{x}_{t,i}) - \mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]| \\
&= \left| \mathbb{E}\left[\frac{1}{|\mathcal{E}_{f,p_{t,i}}(t)|} \sum_{d \in \mathcal{E}_{f,p_{t,i}}(t)} \big(\mu_f(\mathbf{x}_{t,i}) - d\big) \right] \right| \\
&= \left| \mathbb{E}\left[\mathbb{E}\left[\frac{1}{|\mathcal{E}_{f,p_{t,i}}(t)|} \sum_{d \in \mathcal{E}_{f,p_{t,i}}(t)} \big(\mu_f(\mathbf{x}_{t,i}) - d\big) \Big| \mathcal{E}_{f,p_{t,i}}(t) \right] \right] \right| \\
&= \left| \mathbb{E}\left[\frac{1}{|\mathcal{E}_{f,p_{t,i}}(t)|} \sum_{d \in \mathcal{E}_{f,p_{t,i}}(t)} \Big(\mu_f(\mathbf{x}_{t,i}) - \mathbb{E}\big[d \big| \mathcal{E}_{f,p_{t,i}}(t)\big] \Big) \right] \right| \\
&\leq \mathbb{E}\left[\frac{1}{|\mathcal{E}_{f,p_{t,i}}(t)|} \sum_{d \in \mathcal{E}_{f,p_{t,i}}(t)} L \left|\left| \left[\frac{1}{h_T}, \ldots, \frac{1}{h_T}\right] \right|\right|_D^{\alpha} \right] \\
&\leq L D^{\frac{\alpha}{2}} h_T^{-\alpha}, \qquad \text{(A.30)}
\end{aligned}
$$

where we used the definition of $\hat{\mu}_{f,p_{t,i}}(t)$ and the linearity of expectation in the first line and the law of total expectation in the second line [BW16]. In the third line, we used the property of conditional expectation which allows to pull known factors out of the conditional expectation [BW16]. In the fourth line, we used the triangle inequality and since the corresponding context of each of the observed demands $d \in \mathcal{E}_{f,p_{t,i}}(t)$ came from hypercube $p_{t,i}$, we used the Hölder continuity from Assumption 4.1 and exploited the size $\frac{1}{h_T} \times \ldots \times \frac{1}{h_T}$ of the hypercubes. Hence, for the first term in (A.27), by first

using (A.30), then (A.29) and then (A.28), it follows that

$$
\begin{aligned}
&I_{\{V(t)\}} \cdot \left(\sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \Big(w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} \mu_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)}(\mathbf{x}_{t,i}) - w_{c_{t,j}} \mu_{c_{t,j}}(\mathbf{x}_{t,i}) \Big) \right) \\
&\leq I_{\{V(t)\}} \cdot \left(\sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \Bigg(w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} \, \mathbb{E}[\hat{\mu}_{f_j^*(\mathcal{X}_t,\mathcal{G}_t),p_{t,i}}(t)] - w_{c_{t,j}} \, \mathbb{E}[\hat{\mu}_{c_{t,j},p_{t,i}}(t)] \right. \\
&\quad \left. + w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} L D^{\frac{\alpha}{2}} h_T^{-\alpha} + w_{c_{t,j}} L D^{\frac{\alpha}{2}} h_T^{-\alpha} \Bigg) \right) \\
&\leq I_{\{V(t)\}} \cdot \left(\sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \Bigg(w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} \hat{\mu}_{f_j^*(\mathcal{X}_t,\mathcal{G}_t),p_{t,i}}(t) - w_{c_{t,j}} \hat{\mu}_{c_{t,j},p_{t,i}}(t) \right. \\
&\quad + w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} H(t) + w_{c_{t,j}} H(t) \\
&\quad \left. + w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} L D^{\frac{\alpha}{2}} h_T^{-\alpha} + w_{c_{t,j}} L D^{\frac{\alpha}{2}} h_T^{-\alpha} \Bigg) \right) \\
&\leq \sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \Bigg(w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} H(t) + w_{c_{t,j}} H(t) \\
&\quad + w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} L D^{\frac{\alpha}{2}} h_T^{-\alpha} + w_{c_{t,j}} L D^{\frac{\alpha}{2}} h_T^{-\alpha} \Bigg)
\end{aligned} \tag{A.31}
$$

holds almost surely. Taking the expectation of (A.27) and exploiting that (A.31) holds almost surely for any $t \in \tilde{\tau}_T$ under $V(t)$ yields

$$
\begin{aligned}
&\mathbb{E}[R_{\text{oi}}(T)] \\
&\leq \sum_{t=1}^{T} \sum_{j=1}^{m} \sum_{i=1}^{U_t} v_{g_{t,i}} \Big((w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} + w_{c_{t,j}}) H(t) + (w_{f_j^*(\mathcal{X}_t,\mathcal{G}_t)} + w_{c_{t,j}}) L D^{\frac{\alpha}{2}} h_T^{-\alpha} \Big) \\
&\quad + \mathbb{E}\left[\sum_{t \in \tilde{\tau}_T} m U_{\max} v_{\max} w_{\max} R_{\max} I_{\{V^c(t)\}} \right].
\end{aligned} \tag{A.32}
$$

Finally, using that the service weights are bounded by $v_{\max}$ for the different service groups and by $w_{\max}$ for the prioritization weights and using $h_T^{-\alpha} = \lceil T^{\gamma} \rceil^{-\alpha} \leq T^{-\alpha\gamma}$, we further have

$$
\begin{aligned}
\mathbb{E}[R_{\text{oi}}(T)] \leq \sum_{t=1}^{T} 2 m U_{\max} v_{\max} w_{\max} \Big(H(t) + L D^{\frac{\alpha}{2}} T^{-\alpha\gamma} \Big) \\
+ \mathbb{E}\left[\sum_{t \in \tilde{\tau}_T} m U_{\max} v_{\max} w_{\max} R_{\max} I_{\{V^c(t)\}} \right].
\end{aligned} \tag{A.33}
$$

Next, we take care of the term with the expected value in (A.33). We can write

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} mU_{\max}v_{\max}w_{\max}R_{\max}I_{\{V^c(t)\}}\right] \\
&= mU_{\max}v_{\max}w_{\max}R_{\max}\,\mathbb{E}\left[\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} I_{\{V^c(t)\}}\middle|\tilde{\tau}_T\right]\right] \\
&= mU_{\max}v_{\max}w_{\max}R_{\max}\,\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} \mathbb{E}\left[I_{\{V^c(t)\}}\middle|\tilde{\tau}_T\right]\right] \\
&= mU_{\max}v_{\max}w_{\max}R_{\max}\,\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} \Pr\big(V^c(t)\big|\tilde{\tau}_T\big)\right], \qquad \text{(A.34)}
\end{aligned}$$

where we used the law of total expectation and the property of conditional expectation which allows to pull known factors out of the conditional expectation [BW16].

Next, we bound $\Pr(V^c(t)|\tilde{\tau}_T)$ for $t \in \tilde{\tau}_T$. The event $V^c(t)$ can be written as

$$V^c(t) = \{\exists f \in \mathcal{F}, i \in \{1,...,U_t\} \text{ s.t. } |\hat{\mu}_{f,p_{t,i}}(t) - \mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]| \geq H(t)\}. \qquad \text{(A.35)}$$

Hence,

$$\begin{aligned}
&\Pr(V^c(t)|\tilde{\tau}_T) \\
&= \Pr(\exists f \in \mathcal{F}, i \in \{1,...,U_t\} \text{ s.t. } |\hat{\mu}_{f,p_{t,i}}(t) - \mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]| \geq H(t)|\tilde{\tau}_T) \\
&\leq \sum_{f\in\mathcal{F}}\sum_{i=1}^{U_t} \Pr(|\hat{\mu}_{f,p_{t,i}}(t) - \mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]| \geq H(t)|\tilde{\tau}_T). \qquad \text{(A.36)}
\end{aligned}$$

For $t \in \tilde{\tau}_T$, we get by the definition of $\mathcal{F}_t^{\text{ue}}$, that $N_{f,p_{t,i}}(t) > K(t) = t^z \log(t)$ holds for each $f \in \mathcal{F}$ and each $i = 1,...,U_t$, and hence, $|\mathcal{E}_{f,p_{t,i}}(t)| > t^z \log(t)$. For $f \in \mathcal{F}$, $i = 1,...,U_t$, and $t \in \tilde{\tau}_T$, applying Hoeffding's inequality [Hoe63] and using $|\mathcal{E}_{f,p_{t,i}}(t)| > t^z \log(t)$, we get

$$\begin{aligned}
&\Pr\big(|\hat{\mu}_{f,p_{t,i}}(t) - \mathbb{E}[\hat{\mu}_{f,p_{t,i}}(t)]| \geq H(t)|\tilde{\tau}_T\big) \\
&\leq 2\exp\left(-2H(t)^2 t^z \log(t)\frac{1}{R_{\max}^2}\right). \qquad \text{(A.37)}
\end{aligned}$$

Hence, the regret due to exploitation phases is bounded by

$$\begin{aligned}
&\mathbb{E}[R_{\text{oi}}(T)] \\
&\leq \sum_{t=1}^{T} 2mU_{\max}v_{\max}w_{\max}\left(H(t) + LD^{\frac{\alpha}{2}}T^{-\alpha\gamma}\right) \\
&\quad + mU_{\max}v_{\max}w_{\max}R_{\max}\,\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T}\sum_{f\in\mathcal{F}}\sum_{i=1}^{U_t} 2\exp\left(-2H(t)^2t^z\log(t)\frac{1}{R_{\max}^2}\right)\right] \\
&\leq \sum_{t=1}^{T} 2mU_{\max}v_{\max}w_{\max}\left(H(t) + LD^{\frac{\alpha}{2}}T^{-\alpha\gamma}\right) \\
&\quad + mU_{\max}v_{\max}w_{\max}R_{\max}\sum_{t=1}^{T}\sum_{f\in\mathcal{F}}\sum_{i=1}^{U_t} 2\exp\left(-2H(t)^2t^z\log(t)\frac{1}{R_{\max}^2}\right). \qquad \text{(A.38)}
\end{aligned}$$

So far, the analysis was performed with respect to an arbitrary $H(t) > 0$. Setting $H(t) := R_{\max}t^{-\frac{z}{2}}$, we get

$$\begin{aligned}
&\mathbb{E}[R_{\text{oi}}(T)] \\
&\leq \sum_{t=1}^{T} 2mU_{\max}v_{\max}w_{\max}\left(R_{\max}t^{-\frac{z}{2}} + LD^{\frac{\alpha}{2}}T^{-\alpha\gamma}\right) \\
&\quad + mU_{\max}v_{\max}w_{\max}R_{\max}\sum_{t=1}^{T}\sum_{f\in\mathcal{F}}\sum_{i=1}^{U_t} 2\exp\left(\frac{-2R_{\max}^2(t^{-\frac{z}{2}})^2t^z\log(t)}{R_{\max}^2}\right) \\
&\leq 2mU_{\max}v_{\max}w_{\max}R_{\max}\sum_{t=1}^{T} t^{-\frac{z}{2}} + 2mU_{\max}v_{\max}w_{\max}\sum_{t=1}^{T} LD^{\frac{\alpha}{2}}T^{-\alpha\gamma} \\
&\quad + mU_{\max}^2v_{\max}w_{\max}R_{\max}|\mathcal{F}|\sum_{t=1}^{T} 2t^{-2} \\
&\leq 2mU_{\max}v_{\max}w_{\max}R_{\max}\frac{T^{1-\frac{z}{2}}}{1-\frac{z}{2}} + 2mU_{\max}v_{\max}w_{\max}LD^{\frac{\alpha}{2}}T^{1-\alpha\gamma} \\
&\quad + mU_{\max}^2v_{\max}w_{\max}R_{\max}|\mathcal{F}|\frac{\pi^2}{3}, \qquad \text{(A.39)}
\end{aligned}$$

where, in the last step, we used the result from (A.74) in Appendix A.9 and the value of the Dirichlet series [GR14]. □

The overall regret in (A.15) can now be bounded by applying Lemmas A.1 and A.2.

Proof of Theorem 4.1. First, let $K(t) = t^z\log(t)$, $t = 1, ..., T$, and $h_T = \lceil T^\gamma \rceil$, where $0 < z < 1$ and $0 < \gamma < \frac{1}{D}$. Then, under Assumption 4.1, by combining

the results of Lemmas A.1 and A.2, the regret $R(T)$ is bounded by

$$\begin{aligned} R(T) \leq\ & mU_{\max}v_{\max}w_{\max}R_{\max}2^D|\mathcal{F}| \cdot (\log(T)T^{z+\gamma D} + T^{\gamma D}) \\ & + 2mU_{\max}v_{\max}w_{\max}R_{\max}\frac{T^{1-\frac{z}{2}}}{1-\frac{z}{2}} + 2mU_{\max}v_{\max}w_{\max}LD^{\frac{\alpha}{2}}T^{1-\alpha\gamma} \\ & + mU_{\max}^2v_{\max}w_{\max}R_{\max}|\mathcal{F}|\frac{\pi^2}{3} \end{aligned} \tag{A.40}$$

The summands contribute to the regret with leading orders $O(T^{z+\gamma D}\log(T))$, $O(T^{1-\frac{z}{2}})$ and $O(T^{1-\alpha\gamma})$. We balance the leading orders by setting the parameters z and γ according to $z := \frac{2\alpha}{3\alpha+D} \in (0,1)$, $\gamma := \frac{z}{2\alpha} \in (0, \frac{1}{D})$. Then, the regret $R(T)$ is bounded by

$$\begin{aligned} R(T) \leq\ & mU_{\max}v_{\max}w_{\max}R_{\max}2^D|\mathcal{F}| \cdot (\log(T)T^{\frac{2\alpha+D}{3\alpha+D}} + T^{\frac{D}{3\alpha+D}}) \\ & + \frac{2mU_{\max}v_{\max}w_{\max}R_{\max}}{(2\alpha+D)/(3\alpha+D)}T^{\frac{2\alpha+D}{3\alpha+D}} \\ & + 2mU_{\max}v_{\max}w_{\max}LD^{\frac{\alpha}{2}}T^{\frac{2\alpha+D}{3\alpha+D}} + mU_{\max}^2v_{\max}w_{\max}R_{\max}|\mathcal{F}|\frac{\pi^2}{3}. \end{aligned} \tag{A.41}$$

The leading order of the regret is hence $O\left(\log(T)T^{\frac{2\alpha+D}{3\alpha+D}}\right)$. □

A.6 Proof of Theorem 4.2

In this appendix, we prove Theorem 4.2 from Section 4.7.2, yielding an upper bound on the regret of CAC for rating-based caching with missing ratings.

Proof of Theorem 4.2. The proof of Theorem 4.2 works analogously to the proof of Theorem 4.1 (applied to $\tilde{d}_f(\mathbf{x})$ instead of $d_f(\mathbf{x})$), by first dividing the regret into two summands and bounding each summand. The only difference in CAC is that the counters are not updated if no ratings are given for requested files. Since the control function remains the same, the regret due to exploitation phases is not influenced by missing ratings. Therefore, the only difference occurs in the proof of the regret due to exploration phases. In the proof of Lemma A.1, it is argued that for each file $f \in \mathcal{F}$, the number of exploration phases, in which f is selected since it is under-explored is bounded above by $\lceil T^z \log(T) \rceil$ for each set in the partition of the context space. However, in case no rating is given within one time slot in which a file was selected and requested, the counters of CAC are not updated. Hence, in this case, the number of required exploration phases increases. Due to the uncertainty of rating revealings, this

number of exploration phases is also uncertain. Next, an upper bound on the expected value of this number is given.

Consider a fixed file $f \in \mathcal{F}$ and a fixed time slot t. The number U_t of users in time slot t lies between 1 and $U_{\max}$. With probability β, a user reveals her/his rating after requesting file f. Assume that among the U_t users, a number $1 \leq l_t \leq U_t$ of them requests file f in time slot t. (Note that in the case that *no* user requests file f, the counters in CAC are updated and hence, this case does not increase the number of exploration steps.) Then, the probability of not receiving any rating from these l_t users in time slot t is given by $(1-\beta)^{l_t}$. Moreover, this probability is maximal in case $l_t = 1$, i.e., when only one user may potentially rate file f. In the following, we consider this worst case, that exactly one user requests file f and may potentially rate file f in any relevant time slot t. Consider a fixed hypercube $p \in \mathcal{P}_T$ of the partition. Let $\tau_{f,p}(T)$ be the number of exploration phases until there are $\lceil T^z \log(T) \rceil$ ratings for file f revealed in hypercube p. Then, the expected number of exploration phases for file f in hypercube p at the time horizon T is given by $\mathbb{E}[\tau_{f,p}(T)]$. Let $X_{f,p}(y)$ be the number of time slots, in which file f is cached, the context of the user requesting file f comes from hypercube p, but the user does not give a rating for file f, until y ratings have been given. Then,

$$\mathbb{E}[\tau_{f,p}(T)] = \mathbb{E}\left[X_{f,p}(\lceil T^z \log(T) \rceil)\right] + \lceil T^z \log(T) \rceil \tag{A.42}$$

holds. The random variable $X_{f,p}(y)$ has negative binomial distribution [DS12] with a probability of $1-\beta$ that no rating is revealed. Hence, its expected value at $\lceil T^z \log(T) \rceil$ is

$$\mathbb{E}\left[X_{f,p}(\lceil T^z \log(T) \rceil)\right] = \frac{(1-\beta)\lceil T^z \log(T) \rceil}{\beta}. \tag{A.43}$$

Therefore,

$$\mathbb{E}\left[\tau_{f,p}(T)\right] = \frac{1}{\beta}\lceil T^z \log(T) \rceil. \tag{A.44}$$

Hence, going back to the general case (in which $0 \leq l_t \leq U_t$ holds for the number l_t of users requesting file f in time slot t), an upper bound on the expected number of time slots of exploration phases, in which a file f is selected for any of the $(h_T)^D$ hypercubes of the partition is given by $\frac{1}{\beta}(h_T)^D \lceil T^z \log(T) \rceil$. The remainder of the proof works as in the proof of Lemma A.1 given in Appendix A.5. □

A.7 Proof of Theorem 5.1

In this appendix, we prove Theorem 5.1 from Section 5.6.1, showing that the regret of HCL is sublinear in T, which guarantees that HCL converges to the centralized oracle

solution for $T \to \infty$. First, we prove three lemmas and then, we combine the results of the three lemmas to conclude the proof.

Given an arbitrary length T sequence of task and worker arrivals, let τ_T be the set of tasks in $\{1, ..., T\}$ for which $W_t > m_t$, and $\tau_T^c = \{1, ..., T\} \setminus \tau_T$. τ_T^c is also called the set of select-all-workers phases. Also let $\tilde{\tau}_T \subseteq \tau_T$ be the set of tasks in τ_T for which the MCSP is in exploitation phase, and $\tilde{\tau}_T^c = \tau_T \setminus \tilde{\tau}_T$ be the set of tasks in τ_T for which the MCSP is in exploration phase. $\tilde{\tau}_T$ and $\tilde{\tau}_T^c$ are random sets that depend on the selections of the MCSP and the randomness of the observed performances. Let $R_{\text{all}}(T)$, $R_{\text{or}}(T)$ and $R_{\text{oi}}(T)$ represent the regret due to select-all-workers phases, due to exploration phases and due to exploitation phases, respectively. Using the expressions above, the regret $R(T)$ in (5.10) can be decomposed as follows:

$$R(T) = \mathbb{E}\left[R_{\text{all}}(T) + R_{\text{or}}(T) + R_{\text{oi}}(T)\right], \tag{A.45}$$

where

$$R_{\text{all}}(T) := \sum_{t \in \tau_T^c} \sum_{j=1}^{\min\{m_t, W_t\}} \left(\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t)\right) \tag{A.46}$$

$$R_{\text{or}}(T) := \sum_{t \in \tilde{\tau}_T^c} \sum_{j=1}^{\min\{m_t, W_t\}} \left(\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t)\right) \tag{A.47}$$

$$R_{\text{oi}}(T) := \sum_{t \in \tilde{\tau}_T} \sum_{j=1}^{\min\{m_t, W_t\}} \left(\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t)\right). \tag{A.48}$$

The regret is computed by considering the loss due to selecting workers $\{s_{t,j}\}_{j=1,...,\min\{m_t,W_t\}}$ instead of the optimal workers $\{s^*_{t,j}\}_{j=1,...,\min\{m_t,W_t\}}$ for each task. This loss is computed by subtracting the sum of expected performances of the optimal workers from the sum of expected performances of the selected workers.

Next, we will bound the expected values of each of the three summands above separately. First, we show that the regret due to select-all-workers phases is 0.

Lemma A.3 (Value of $\mathbb{E}[R_{\text{all}}(T)]$)**.** *When LC i, $i \in \mathcal{W}$, runs Algorithm 5.1 with an arbitrary deterministic function $K_i : \{1, ..., T\} \to \mathbb{R}_+$ and an arbitrary $h_{T,i} \in \mathbb{N}$ as input, and the MCSP runs Algorithm 5.2, the regret $\mathbb{E}\left[R_{\text{all}}(T)\right]$ satisfies*

$$\mathbb{E}[R_{\text{all}}(T)] = 0. \tag{A.49}$$

Proof of Lemma A.3. For $t \in \tau_T^c$, i.e., $W_t \leq m_t$, the MCSP enters a select-all-workers phase. Moreover, for $W_t \leq m_t$, the trivial optimal solution is to request all available

workers to complete task t. Hence, the MCSP's selection of workers is optimal and therefore, select-all-workers phases do not contribute to the regret, i.e., $\mathbb{E}[R_{\text{all}}(T)] = 0$. □

Next, a bound for $\mathbb{E}[R_{\text{or}}(T)]$ is given.

Lemma A.4 (Bound for $\mathbb{E}[R_{\text{or}}(T)]$)**.** *When LC i, $i \in \mathcal{W}$, runs Algorithm 5.1 with input parameters $K_i(t) = t^{z_i}\log(t)$, $t = 1, ..., T$, and $h_{T,i} = \lceil T^{\gamma_i} \rceil$, where $0 < z_i < 1$ and $0 < \gamma_i < \frac{1}{D_i}$, and the MCSP runs Algorithm 5.2, the regret $\mathbb{E}[R_{\text{or}}(T)]$ is bounded by*

$$\mathbb{E}[R_{\text{or}}(T)] \leq W q_{\max} \sum_{i \in \mathcal{W}} 2^{D_i} (\log(T) T^{z_i + \gamma_i D_i} + T^{\gamma_i D_i}). \tag{A.50}$$

Proof of Lemma A.4. Let $t \in \tilde{\tau}_T^c$ be a task for which the MCSP enters an exploration phase. By design of HCL, in this case, $W_t > m_t$ holds, i.e., $m_t = \min\{m_t, W_t\}$. Since the expected performance of a worker is bounded in $[0, q_{\max}]$, it follows that

$$\begin{aligned} R_{\text{or}}(T) &= \sum_{t \in \tilde{\tau}_T^c} \sum_{j=1}^{m_t} \left(\theta_{s_{t,j}^*}(\mathbf{x}_{t,s_{t,j}^*}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t) \right) \\ &\leq \sum_{t \in \tilde{\tau}_T^c} m_t q_{\max}. \end{aligned} \tag{A.51}$$

Hence, the regret can be bounded by

$$\begin{aligned} \mathbb{E}[R_{\text{or}}(T)] &\leq \mathbb{E}\left[\sum_{t \in \tilde{\tau}_T^c} m_t q_{\max} \right] \\ &\leq W q_{\max} \, \mathbb{E}\left[\sum_{t \in \tilde{\tau}_T^c} 1 \right], \end{aligned} \tag{A.52}$$

since $m_t \leq W$ holds for all $t = 1, ..., T$.

For $t \in \tilde{\tau}_T^c$, the set $\mathcal{W}_t^{\text{ue}}$ of under-explored workers is non-empty. Hence, there exists an available worker $i \in \mathcal{W}_t$ with $N_{i,q_{t,i}}(t) \leq K_i(t) = t^{z_i}\log(t)$. By definition of $\mathcal{W}_t^{\text{ue}}$, up to task T, worker i can induce at most $\lceil T^{z_i}\log(T) \rceil$ exploration phases for each of the $(h_{T,i})^{D_i}$ hypercubes of the partition $\mathcal{Q}_{T,i}$. Hence, the number of exploration phases is upper-bounded as follows:

$$\mathbb{E}\left[\sum_{t \in \tilde{\tau}_T^c} 1 \right] \leq \sum_{i \in \mathcal{W}} (h_{T,i})^{D_i} \lceil T^{z_i} \log(T) \rceil. \tag{A.53}$$

This upper bound is rather loose as several workers might be explored simultaneously, in which case they do not induce separate exploration phases. From (A.53), we conclude

$$\mathbb{E}[R_{\text{or}}(T)] \leq W q_{\max} \sum_{i \in \mathcal{W}} (h_{T,i})^{D_i} \lceil T^{z_i} \log(T) \rceil. \tag{A.54}$$

Using $(h_{T,i})^{D_i} = \lceil T^{\gamma_i} \rceil^{D_i} \leq (2T^{\gamma_i})^{D_i} = 2^{D_i} T^{\gamma_i D_i}$, we find

$$\mathbb{E}[R_{\text{or}}(T)] \leq W q_{\max} \sum_{i \in \mathcal{W}} 2^{D_i} (\log(T) T^{z_i + \gamma_i D_i} + T^{\gamma_i D_i}). \tag{A.55}$$

□

Next, we give a bound for $\mathbb{E}\left[R_{\text{oi}}(T)\right]$.

Lemma A.5 (Bound for $\mathbb{E}\left[R_{\text{oi}}(T)\right]$)**.** *Given that Assumption 5.1 from Section 5.6.1 holds, when LC i, $i \in \mathcal{W}$, runs Algorithm 5.1 with parameters $K_i(t) = t^{z_i} \log(t)$, $t = 1, ..., T$, and $h_{T,i} = \lceil T^{\gamma_i} \rceil$, where $0 < z_i < 1$ and $0 < \gamma_i < \frac{1}{D_i}$, and the MCSP runs Algorithm 5.2, the regret $\mathbb{E}\left[R_{\text{oi}}(T)\right]$ is bounded by*

$$\begin{aligned} \mathbb{E}[R_{\text{oi}}(T)] \leq 2 \sum_{i \in \mathcal{W}} q_{\max} \frac{T^{1 - \frac{z_i}{2}}}{1 - \frac{z_i}{2}} + 2 \sum_{i \in \mathcal{W}} L D_i{}^{\frac{\alpha}{2}} T^{1 - \alpha \gamma_i} \\ + q_{\max} W^2 \frac{\pi^2}{3}. \end{aligned} \tag{A.56}$$

Proof of Lemma A.5. Let $t \in \tilde{\tau}_T$, i.e., the MCSP enters an exploitation phase. By design of HCL, in this case, $W_t > m_t$ holds, i.e., $m_t = \min\{m_t, W_t\}$. Additionally, since in exploitation phases, the set of under-explored workers is empty (i.e., $\mathcal{W}_t^{\text{ue}} = \emptyset$), $N_{i,q_{t,i}}(t) > K_i(t) = t^{z_i} \log(t)$ holds for all available workers $i \in \mathcal{W}_t$.

Now, let $V(t)$ be the event that at the arrival of task t, each available worker i's estimated performance $\hat{\theta}_{i,q_{t,i}}(t)$ in the current hypercube $q_{t,i}$ is "close" to its true expected value $\mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]$, i.e.,

$$V(t) = \{|\hat{\theta}_{i,q_{t,i}}(t) - \mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]| < H_i(t) \text{ for all } i \in \mathcal{W}_t\} \tag{A.57}$$

for arbitrary $H_i(t) > 0$, $i \in \mathcal{W}_t$. Next, we distinguish between exploitation phases in which $V(t)$ or its complementary event, denoted by $V^c(t)$, hold. Let $I_{\{.\}}$ denote the indicator function. Then, we can write

$$\begin{aligned} & R_{\text{oi}}(T) \\ & = \sum_{t \in \tilde{\tau}_T} \left(I_{\{V(t)\}} \left(\sum_{j=1}^{m_t} (\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t)) \right) \right) \\ & + \sum_{t \in \tilde{\tau}_T} \left(I_{\{V^c(t)\}} \left(\sum_{j=1}^{m_t} (\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t)) \right) \right). \end{aligned} \tag{A.58}$$

Using that the expected performance of a worker is bounded in $[0, q_{\max}]$, this term can further be bounded as

$$\begin{aligned} & R_{\text{oi}}(T) \\ & \leq \sum_{t \in \tilde{\tau}_T} \left(I_{\{V(t)\}} \cdot \left(\sum_{j=1}^{m_t} (\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t)) \right) \right) \\ & \quad + \sum_{t \in \tilde{\tau}_T} m_t q_{\max} I_{\{V^c(t)\}}. \end{aligned} \tag{A.59}$$

First, we bound the first term in (A.59). We start by noting that in an exploitation phase $t \in \tilde{\tau}_T$, since the MCSP selected workers $\{s_{t,j}\}_{j=1,\dots,m_t}$ instead of $\{s^*_{t,j}\}_{j=1,\dots,m_t}$, we have

$$\sum_{j=1}^{m_t} \hat{\theta}_{s^*_{t,j}, q_{s^*_{t,j}}}(t) \leq \sum_{j=1}^{m_t} \hat{\theta}_{s_{t,j}, q_{s_{t,j}}}(t). \tag{A.60}$$

We also know that when $V(t)$ holds, we have

$$\{|\hat{\theta}_{i,q_{t,i}}(t) - \mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]| < H_i(t) \text{ for all } i \in \mathcal{W}_t\} \tag{A.61}$$

almost surely. Finally, note that by the Hölder continuity from Assumption 5.1, since $(\mathbf{x}_{t,i}, \mathbf{c}_t) \in q_{t,i}$ and for calculating $\hat{\theta}_{i,q_{t,i}}(t)$, only contexts from hypercube $q_{t,i}$ are used, for each $i \in \mathcal{W}_t$, it follows that

$$\begin{aligned} & |\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t) - \mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]| \\ & = \left| \mathbb{E}\left[\frac{1}{|\mathcal{E}_{i,q_{t,i}}(t)|} \sum_{p \in \mathcal{E}_{i,q_{t,i}}(t)} \big(\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t) - p\big) \right] \right| \\ & = \left| \mathbb{E}\left[\mathbb{E}\left[\frac{1}{|\mathcal{E}_{i,q_{t,i}}(t)|} \sum_{p \in \mathcal{E}_{i,q_{t,i}}(t)} \big(\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t) - p\big) \Big| \mathcal{E}_{i,q_{t,i}}(t) \right] \right] \right| \\ & = \left| \mathbb{E}\left[\frac{1}{|\mathcal{E}_{i,q_{t,i}}(t)|} \sum_{p \in \mathcal{E}_{i,q_{t,i}}(t)} \Big(\theta_i(\mathbf{x}_{t,i}, \mathbf{c}_t) - \mathbb{E}\big[p \big| \mathcal{E}_{i,q_{t,i}}(t)\big] \Big) \right] \right| \\ & \leq \mathbb{E}\left[\frac{1}{|\mathcal{E}_{i,q_{t,i}}(t)|} \sum_{p \in \mathcal{E}_{i,q_{t,i}}(t)} L \Big| \Big| \big(\frac{1}{h_{T,i}}, \dots, \frac{1}{h_{T,i}}\big) \Big| \Big|^{\alpha}_{D_i} \right] \\ & \leq L D_i^{\frac{\alpha}{2}} h_{T,i}^{-\alpha}, \end{aligned} \tag{A.62}$$

where we used the definition of $\hat{\theta}_{i,q_{t,i}}(t)$ and the linearity of expectation in the first line and the law of total expectation in the second line. In the third line, we used the property of conditional expectation which allows to pull known factors out of the conditional expectation [BW16]. In the fourth line, we used the triangle inequality and

since the corresponding context of each of the observed performances $p \in \mathcal{E}_{i,q_{t,i}}(t)$ came from hypercube $q_{t,i}$, we used the Hölder continuity from Assumption 5.1 and exploited the size $\frac{1}{h_{T,i}} \times \ldots \times \frac{1}{h_{T,i}}$ of the hypercubes. Hence, by first using (A.62), then (A.61) and then (A.60), we have for the first term in (A.59) that

$$
\begin{aligned}
& I_{\{V(t)\}} \cdot \left(\sum_{j=1}^{m_t} \left(\theta_{s^*_{t,j}}(\mathbf{x}_{t,s^*_{t,j}}, \mathbf{c}_t) - \theta_{s_{t,j}}(\mathbf{x}_{t,s_{t,j}}, \mathbf{c}_t) \right) \right) \\
& \leq I_{\{V(t)\}} \cdot \left(\sum_{j=1}^{m_t} \left(\mathbb{E}[\hat{\theta}_{s^*_{t,j},q_{s^*_{t,j}}}(t)] - \mathbb{E}[\hat{\theta}_{s_{t,j},q_{s_{t,j}}}(t)] \right. \right. \\
& \quad \left. \left. + LD_{s^*_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s^*_{t,j}} + LD_{s_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s_{t,j}} \right) \right) \\
& \leq I_{\{V(t)\}} \cdot \left(\sum_{j=1}^{m_t} \hat{\theta}_{s^*_{t,j},q_{s^*_{t,j}}}(t) - \sum_{j=1}^{m_t} \hat{\theta}_{s_{t,j},q_{s_{t,j}}}(t) \right. \\
& \quad + \sum_{j=1}^{m_t} \left(H_{s^*_{t,j}}(t) + H_{s_{t,j}}(t) \right. \\
& \quad \left. \left. + LD_{s^*_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s^*_{t,j}} + LD_{s_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s_{t,j}} \right) \right) \\
& \leq \sum_{j=1}^{m_t} \left(H_{s^*_{t,j}}(t) + H_{s_{t,j}}(t) \right. \\
& \quad \left. + LD_{s^*_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s^*_{t,j}} + LD_{s_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s_{t,j}} \right) \qquad \text{(A.63)}
\end{aligned}
$$

holds almost surely. Taking the expectation of (A.59) and exploiting that (A.63) holds almost surely for any $t \in \tilde{\tau}_T$ yields

$$
\begin{aligned}
& \mathbb{E}[R_{\text{oi}}(T)] \\
& \leq \sum_{t=1}^{T} \left(\sum_{j=1}^{m_t} (H_{s^*_{t,j}}(t) + H_{s_{t,j}}(t) \right. \\
& \quad \left. + LD_{s^*_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s^*_{t,j}} + LD_{s_{t,j}}{}^{\frac{\alpha}{2}} h^{-\alpha}_{T,s_{t,j}}) \right) \\
& \quad + \mathbb{E}\left[\sum_{t \in \tilde{\tau}_T} m_t q_{\max} I_{\{V^c(t)\}} \right]. \qquad \text{(A.64)}
\end{aligned}
$$

Finally, adding non-negative summands and using $h^{-\alpha}_{T,i} = \lceil T^{\gamma_i} \rceil^{-\alpha} \leq T^{-\alpha\gamma_i}$, we further

have

$$\begin{aligned}
&\mathbb{E}[R_{\mathrm{oi}}(T)] \\
&\leq \sum_{t=1}^{T}\left(2\sum_{i\in\mathcal{W}} H_i(t) + 2\sum_{i\in\mathcal{W}} L{D_i}^{\frac{\alpha}{2}} T^{-\alpha\gamma_i}\right) \\
&\quad + \mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} m_t q_{\max} I_{\{V^c(t)\}}\right]. \qquad \text{(A.65)}
\end{aligned}$$

Next, we take care of the term with the expected value in the last expression. We can write

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} m_t q_{\max} I_{\{V^c(t)\}}\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} m_t q_{\max} I_{\{V^c(t)\}}\middle|\tilde{\tau}_T\right]\right] \\
&= \mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} m_t q_{\max}\,\mathbb{E}\left[I_{\{V^c(t)\}}\middle|\tilde{\tau}_T\right]\right] \\
&= \mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} m_t q_{\max}\Pr\big(V^c(t)\big|\tilde{\tau}_T\big)\right], \qquad \text{(A.66)}
\end{aligned}$$

where we used the law of total expectation and the property of conditional expectation which allows to pull known factors out of the conditional expectation [BW16].

Next, we bound $\Pr(V^c(t)|\tilde{\tau}_T)$ for $t \in \tilde{\tau}_T$. The event $V^c(t)$ can be written as

$$V^c(t) = \{\exists i \in \mathcal{W}_t \text{ s.t. } |\hat{\theta}_{i,q_{t,i}}(t) - \mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]| \geq H_i(t)\}. \qquad \text{(A.67)}$$

Hence,

$$\begin{aligned}
&\Pr(V^c(t)|\tilde{\tau}_T) \\
&= \Pr(\exists i \in \mathcal{W}_t \text{ s.t. } |\hat{\theta}_{i,q_{t,i}}(t) - \mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]| \geq H_i(t)|\tilde{\tau}_T) \\
&\leq \sum_{i\in\mathcal{W}_t} \Pr(|\hat{\theta}_{i,q_{t,i}}(t) - \mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]| \geq H_i(t)|\tilde{\tau}_T). \qquad \text{(A.68)}
\end{aligned}$$

For $t \in \tilde{\tau}_T$, we get by the definition of $\mathcal{W}_t^{\mathrm{ue}}$ that $N_{i,q_{t,i}}(t) > K_i(t) = t^{z_i}\log(t)$ holds for each $i \in \mathcal{W}_t$, and hence, $|\mathcal{E}_{i,q_{t,i}}(t)| > t^{z_i}\log(t)$. For $i \in \mathcal{W}_t$ and $t \in \tilde{\tau}_T$, applying Hoeffding's inequality [Hoe63] and using $|\mathcal{E}_{i,q_{t,i}}(t)| > t^{z_i}\log(t)$, we get

$$\begin{aligned}
&\Pr\big(|\hat{\theta}_{i,q_{t,i}}(t) - \mathbb{E}[\hat{\theta}_{i,q_{t,i}}(t)]| \geq H_i(t)|\tilde{\tau}_T\big) \\
&\leq 2\exp\left(-2H_i(t)^2 t^{z_i}\log(t)\frac{1}{q_{\max}^2}\right). \qquad \text{(A.69)}
\end{aligned}$$

Hence, the regret due to exploitation phases is bounded by

$$\begin{aligned}
&\mathbb{E}[R_{\mathrm{oi}}(T)] \\
&\leq \sum_{t=1}^{T}\Big(2\sum_{i\in\mathcal{W}} H_i(t) + 2\sum_{i\in\mathcal{W}} LD_i{}^{\frac{\alpha}{2}}T^{-\alpha\gamma_i}\Big) \\
&\quad + \mathbb{E}\left[\sum_{t\in\tilde{\tau}_T} m_t q_{\max} \sum_{i\in\mathcal{W}_t} 2\exp\left(-2H_i(t)^2 t^{z_i}\log(t)\frac{1}{q_{\max}^2}\right)\right] \\
&\leq \sum_{t=1}^{T}\Big(2\sum_{i\in\mathcal{W}} H_i(t) + 2\sum_{i\in\mathcal{W}} LD_i{}^{\frac{\alpha}{2}}T^{-\alpha\gamma_i}\Big) \\
&\quad + \sum_{t=1}^{T} m_t q_{\max} \sum_{i\in\mathcal{W}_t} 2\exp\left(-2H_i(t)^2 t^{z_i}\log(t)\frac{1}{q_{\max}^2}\right). \qquad \text{(A.70)}
\end{aligned}$$

So far, the analysis was performed with respect to arbitrary $H_i(t) > 0$, $i \in \mathcal{W}$. Setting $H_i(t) := q_{\max} t^{-\frac{z_i}{2}}$ for $i \in \mathcal{W}$, we get

$$\begin{aligned}
&\mathbb{E}[R_{\mathrm{oi}}(T)] \\
&\leq \sum_{t=1}^{T}\Big(2\sum_{i\in\mathcal{W}} q_{\max} t^{-\frac{z_i}{2}} + 2\sum_{i\in\mathcal{W}} LD_i{}^{\frac{\alpha}{2}}T^{-\alpha\gamma_i}\Big) \\
&\quad + \sum_{t=1}^{T} m_t q_{\max} \sum_{i\in\mathcal{W}_t} 2\exp\left(\frac{-2q_{\max}^2 (t^{-\frac{z_i}{2}})^2 t^{z_i}\log(t)}{q_{\max}^2}\right) \\
&\leq 2\sum_{i\in\mathcal{W}} q_{\max} \sum_{t=1}^{T} t^{-\frac{z_i}{2}} + 2\sum_{i\in\mathcal{W}} LD_i{}^{\frac{\alpha}{2}}T^{1-\alpha\gamma_i} \\
&\quad + q_{\max} W \sum_{t=1}^{T} m_t 2t^{-2} \\
&\leq 2\sum_{i\in\mathcal{W}} q_{\max} \frac{T^{1-\frac{z_i}{2}}}{1-\frac{z_i}{2}} + 2\sum_{i\in\mathcal{W}} LD_i{}^{\frac{\alpha}{2}}T^{1-\alpha\gamma_i} \\
&\quad + q_{\max} W^2 \frac{\pi^2}{3}, \qquad \text{(A.71)}
\end{aligned}$$

where, in the last step, we used the result from (A.74) in Appendix A.9, the fact that $m_t \leq W$ holds and the value of the Dirichlet series [GR14]. □

Applying Lemmas A.3–A.5, the overall regret in (A.45) can be bounded as given below.

Proof of Theorem 5.1. First, for $i \in \mathcal{W}$, let $K_i(t) = t^{z_i}\log(t)$ and $h_{T,i} = \lceil T^{\gamma_i} \rceil$, where $0 < z_i < 1$ and $0 < \gamma_i < \frac{1}{D_i}$. Then, under Assumption 5.1, by combining

the results of Lemmas A.3–A.5, the regret $R(T)$ is bounded by

$$\begin{aligned} R(T) \leq q_{\max} W \sum_{i \in \mathcal{W}} 2^{D_i} (\log(T) T^{z_i + \gamma_i D_i} + T^{\gamma_i D_i}) \\ + 2 \sum_{i \in \mathcal{W}} q_{\max} \frac{T^{1 - \frac{z_i}{2}}}{1 - \frac{z_i}{2}} + 2 \sum_{i \in \mathcal{W}} L D_i^{\frac{\alpha}{2}} T^{1 - \alpha \gamma_i} \\ + q_{\max} W^2 \frac{\pi^2}{3}. \end{aligned} \tag{A.72}$$

The summands contribute to the regret with leading orders $O(\sum_{i \in \mathcal{W}} T^{z_i + \gamma_i D_i} \log(T))$, $O(\sum_{i \in \mathcal{W}} T^{1 - \alpha \gamma_i})$ and $O(\sum_{i \in \mathcal{W}} T^{1 - \frac{z_i}{2}})$. We balance the leading orders by setting the parameters z_i, γ_i according to $z_i := \frac{2\alpha}{3\alpha + D_i} \in (0, 1)$, $\gamma_i := \frac{z_i}{2\alpha} \in (0, \frac{1}{D_i})$ for $i \in \mathcal{W}$. Then, the regret $R(T)$ is bounded by

$$\begin{aligned} R(T) \leq q_{\max} W \sum_{i \in \mathcal{W}} 2^{D_i} (\log(T) T^{\frac{2\alpha + D_i}{3\alpha + D_i}} + T^{\frac{D_i}{3\alpha + D_i}}) \\ + \sum_{i \in \mathcal{W}} \frac{2 q_{\max}}{(2\alpha + D_i)/(3\alpha + D_i)} T^{\frac{2\alpha + D_i}{3\alpha + D_i}} \\ + 2 \sum_{i \in \mathcal{W}} L D_i^{\frac{\alpha}{2}} T^{\frac{2\alpha + D_i}{3\alpha + D_i}} + q_{\max} W^2 \frac{\pi^2}{3}. \end{aligned} \tag{A.73}$$

Setting $D_{\max} := \max_{i \in \mathcal{W}} D_i$, the leading order of the regret is hence $O\left(T^{\frac{2\alpha + D_{\max}}{3\alpha + D_{\max}}} \log(T)\right)$. □

A.8 Proof of Corollary 5.1

In this appendix, we prove Corollary 5.1 from Section 5.6.5, giving an upper bound on the number of quality assessments per worker needed in the proposed algorithm HCL.

Proof of Corollary 5.1. This follows directly from the proof of Lemma A.4 given in Appendix A.7 and the proof of Theorem 5.1 given in Appendix A.7. A quality assessment is only requested if a worker is selected for exploration purposes. From the proof of Lemma A.4, the number of times a worker can at most be selected for exploration purposes is upper-bounded by $(h_{T,i})^{D_i} \lceil T^{z_i} \log(T) \rceil = \lceil T^{\gamma_i} \rceil^{D_i} \lceil T^{z_i} \log(T) \rceil \leq (1 + T^{\gamma_i})^{D_i} (1 + T^{z_i} \log(T))$. Setting the parameters z_i, γ_i as in the proof of Theorem 5.1 concludes the proof. □

A.9 A Bound On Divergent Series

The following bound on divergent series is needed in the proof of Theorem 4.1 given in Appendix A.5 and in the proof of Theorem 5.1 given in Appendix A.7.

For $p > 0$, $p \neq 1$, the following formula holds:

$$\sum_{t=1}^{T} \frac{1}{t^p} \leq 1 + \frac{T^{1-p} - 1}{1 - p} \tag{A.74}$$

Proof. See [Chl09]. □

List of Acronyms

AP	Access Point
AUER	Awake Upper Estimated Reward Algorithm from [KNMS10]
CAC	Context-Aware Proactive Caching Algorithm
CACao	Context-Aware Proactive Caching with Area Overlap Algorithm
C-RAN	Cloud Radio Access Network
CS	Crowdsourcing
FDMA	Frequency-Division Multiple Access
HCL	Hierarchical Context-Aware Learning Algorithm
i.i.d.	Independent Identically Distributed
ILP	Integer Linear Programming
IoT	Internet of Things
LC	Local Controller
LFU	Least Frequently Used Algorithm
LinUCB	UCB-type Algorithm for Contextual Bandits with Linear Payoff Functions from [LCLS10, CLRS11]
LRU	Least Recently Used Algorithm
M2M	Machine to Machine
MAB	Multi-Armed Bandit
MBS	Macro Base Station
MCC	Mobile Cloud Computing
MCS	Mobile Crowdsourcing
MCSP	Mobile Crowdsourcing Platform
MEC	Mobile Edge Computing
MIMO	Multiple-Input and Multiple-Output

MNO	Mobile Network Operator
NP	Non-Deterministic Polynomial-Time
OFDMA	Orthogonal Frequency-Division Multiple Access
RL	Reinforcement Learning
SAT	Server Assigned Tasks
SBS	Small Base Station
s.t.	Subject To
TDMA	Time-Division Multiple Access
TR	Task Recommendation
UCB	Upper Confidence Bound
UCB1	Upper Confidence Bound Algorithm from [ACBF02]
w.r.t.	With Respect To
WST	Worker Selected Tasks

List of Mathematical Symbols

$\emptyset$	Empty set
$\lvert s \rvert$	Absolute value of a scalar s
$\lvert \mathcal{S} \rvert$	Cardinality of a set $\mathcal{S}$
$\lceil \cdot \rceil$	Ceiling function, maps a number to the least integer greater than or equal to the number
$\lvert\lvert \cdot \rvert\rvert_D$	Euclidean norm in $\mathbb{R}^D$
$\operatorname{argmax}_x g(x)$	Returns the value of x that maximizes $g(x)$
A^c	Complementary event of an event A
$\mathbb{E}[\cdot]$	Expectation operator
$I_{\{\cdot\}}$	Indicator function
$\mathbb{N}$	Set of positive integer numbers
$O(\cdot)$	O-Notation
$\Pr(\cdot)$	Probability operator
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of positive real numbers
$\mathbb{R}_{0,+}$	Set of non-negative real numbers
$\Omega(\cdot)$	Ω-Notation

List of Variables from Chapter 2

a_t	Action selected in round t
a^*	Optimal action in expectation
$a^*(\mathbf{x}_t)$	Optimal action with respect to context $\mathbf{x}_t$ in expectation
A	Number of actions
$\mathcal{A}$	Set of actions
c	Capacity value
c_j	Capacity value w.r.t. attribute j
d	Number of knapsack constraints
D	Dimension of context space
eff_i	Efficiency of an item i
g_0	Objective function
g_j	Constraint function
I	Number of items
$\mathcal{I}$	Set of items
J	Number of constraint functions
k	Number of optimization variables
L	Parameter in Hölder continuity assumption
$N_a(T)$	Number of times action $a \in \mathcal{A}$ has been played up to T
p_i	Profit of item i
r_{t,a_t}	Reward in round t depending on selected action a_t
$R(T)$	Regret of learning with respect to oracle after T rounds
t	Index of a round
T	Number of rounds
w_i	Weight of item i
$w_{i,j}$	Weight of item i w.r.t. attribute j
$\mathbf{x}_t$	Context arrived in round t
$\mathcal{X}$	Context space
y_i	Binary decision variable for item i
$\mathbf{y}$	Vector of optimization variables
α	Parameter in Hölder continuity assumption
ϵ	Parameter in ϵ-Greedy algorithm
μ_a	Expected reward of action $a \in \mathcal{A}$

List of Variables from Chapter 3

b_n	Bandwidth of node n
B_n	Number of bits of node n's task
$e_{\mathrm{C},n}$	Energy per CPU cycle for local computing at node n
$e_{\mathrm{T},n}$	Energy per bit node n consumes for data transmission
eff_n	Efficiency measure of node n
$E_{\mathrm{C},n}$	Energy consumed by node n if node n computes its task locally
$E_{\mathrm{net}}(\mathbf{y})$	Total energy spent in the network as a function of the actions $\mathbf{y}$
$E_{\mathrm{prov},n}$	Energy provided by node n for a computation offloading session
$E_{\mathrm{T},n}$	Total amount of energy spent in the network if node n uses computation offloading
$E^n_{\mathrm{T},n}$	Energy consumed by node n if node n uses computation offloading
$E^r_{\mathrm{T},n}$	Energy consumed by predecessor $r \in \mathcal{R}_n$ of node n if node n uses computation offloading
h_n	Complex channel coefficient from node n to its parent
L_n	Number of CPU cycles of node n's task
M_n	Processor speed of node n
n	Index of a node
N	Number of nodes
$\mathcal{N}$	Set of nodes
$\mathcal{N}_C$	Set of nodes computing their tasks locally
$\mathcal{N}_T$	Set of nodes transmitting their tasks to the server
$P_{C,n}$	Processing power of node n
$P_{\mathrm{T},n}$	Transmit power $P_{\mathrm{T},n}$ of node n
R	Number of relay nodes
$\mathcal{R}$	Set of relay nodes
$\mathcal{R}_n$	Route from node n to server
y_n	Binary variable describing node n's action
$\mathbf{y}$	Vector of optimization variables
σ_n^2	Noise power at node n

List of Variables from Chapter 4

$c_{t,j}$	j-th cached file in time slot t
$\mathcal{C}^*$	Optimal cache content for sequence of time slots $t = 1, ..., T$
$\mathcal{C}_t$	Set of cached files in time slot t
$\mathcal{C}_t^*(\mathcal{X}_t, \mathcal{G}_t)$	Optimal cache content in time slot t
$d_f(\mathbf{x})$	Random variable for number of times a user with context $\mathbf{x} \in \mathcal{X}$ requests file $f \in \mathcal{F}$ in one time slot
$d_f(\mathbf{x}_{t,i}, t)$	Instantaneous number of times user $i \in \{1, ..., U_t\}$ requests file $f \in \mathcal{F}$ in time slot t
$\tilde{d}_f(\mathbf{x})$	Random variable for combined demand and rating for file f of a user with context $\mathbf{x}$
D	Dimension of context space
$\mathcal{E}_{f,p}(t)$	Set of observed demands of users with context from set $p \in \mathcal{P}_T$ when file $f \in \mathcal{F}$ was cached up to time slot t
$f_j^*(\mathcal{X}_t, \mathcal{G}_t)$	Top-j file for given pair $(\mathcal{X}_t, \mathcal{G}_t)$ of contexts and service groups in time slot t
$\hat{f}_{j,\mathcal{P}_t,\mathcal{G}_t}(t)$	j-th file in file ranking in time slot t
$F_{\mathrm{ue},t}$	Number of under-explored files in time slot t
$\mathcal{F}$	File library
$\mathcal{F}_t^{\mathrm{ue}}$	Set of under-explored files in time slot t
$g_{t,i}$	Service group to which user i in time slot t belongs
$\mathcal{G}$	Set of service groups
$\mathcal{G}_t$	Service groups of all users in time slot t
h_T	Input parameter to Algorithm 4.1, determines number of sets in partition $\mathcal{P}_T$
$H(t)$	Auxiliary function used in regret analysis
$K(t)$	Control function, input to Algorithm 4.1
L	Parameter in Hölder continuity assumption
m	Cache size
$N_{f,p}(t)$	Number of times in which file $f \in \mathcal{F}$ was cached after a user with context from set $p \in \mathcal{P}_T$ was connected to the caching entity before time slot t
o	Overlap parameter for multiple caching entities with area overlap
$p_{t,i}$	Hypercube of partition to which context vector $\mathbf{x}_{t,i}$ belongs
$\mathcal{P}_t$	Set of hypercubes corresponding to set of contexts $\mathcal{X}_t$
$\mathcal{P}_T$	Partition of the context space

$r_{\min}$	Minimum value of user rating
$r_{\max}$	Maximum value of user rating
$r_f(\mathbf{x})$	Random variable for rating of a user with context $\mathbf{x}$ after requesting file f
$R_{\max}$	Maximum possible number of requests a user can submit within one time slot
$R(T)$	Regret of learning with respect to oracle after T rounds
$R_{\mathrm{oi}}(T)$	Regret due to exploitation phases
$R_{\mathrm{or}}(T)$	Regret due to exploration phases
t	Index of a time slot
T	Number of time slots
$U_{\max}$	Maximum number of users that may be connected simultaneously to the caching entity
U_t	Number of users connected to the caching entity in time slot t
v_g	Weight for cache hit by a user of service group $g \in \mathcal{G}$
$v_{\max}$	Maximum weight for cache hit among all service groups
$V(t)$	Auxiliary event used in the regret analysis
w_f	prioritization weight for file $f \in \mathcal{F}$
$w_{\max}$	Maximum prioritization weight among all files in $\mathcal{F}$
$\mathbf{x}_{t,i}$	Context vector of user i in time slot t
$\mathcal{X}$	Context space
$\mathcal{X}_t$	Set of contexts of all users in time slot t
$y_{t,f}$	Binary variable describing if file $f \in \mathcal{F}$ is cached in time slot t
z	Parameter used in regret analysis
α	Parameter in Hölder continuity assumption
β	Probability that a user reveals her/his rating
γ	Parameter used in regret analysis
ϵ	Parameter in ϵ-Greedy algorithm
λ_{CAC}	Factor for control function of Algorithm 4.1 in simulations
$\mu_f(\mathbf{x})$	Expected number of times a user with context $\mathbf{x} \in \mathcal{X}$ requests file $f \in \mathcal{F}$ in one time slot
$\hat{\mu}_{f,p}(t)$	Estimated demand for file $f \in \mathcal{F}$ with respect to set $p \in \mathcal{P}_T$ in time slot t
$\tilde{\tau}_T$	Set of time slots in which CAC enters an exploitation phase, used in regret analysis
$\tilde{\tau}_T^c$	Set of time slots in which CAC enters an exploration phase, used in regret analysis

List of Variables from Chapter 5

$A_i(T)$	Number of quality assessments per worker up to task T
b_t	Budget of task t
$\mathbf{c}_t$	Context of task t
C	Dimension of task context space
$\mathcal{C}$	Task context space
$d_i(\mathbf{x}, \mathbf{c})$	Random variable for decision of worker $i \in \mathcal{W}$ with current personal context $\mathbf{x} \in \mathcal{X}_i$ for a task with task context $\mathbf{c} \in \mathcal{C}$
D_i	Dimension of joint context space of worker $i \in \mathcal{W}$
$D_{\max}$	Maximum of dimensions of joint context spaces
$e_{\max}$	Maximum price to be paid to a worker for completing a task
$e_{\min}$	Minimum price to be paid to a worker for completing a task
e_t	Price to be paid to each worker who completes task t
$\mathcal{E}_{i,q}(t)$	Set of observed performances of worker i before task t when worker i was selected for a task and the joint context was in hypercube q
$\bar{f}_{\mu_i,\sigma_i^2}(c)$	Truncated Gaussian probability density function with mean μ_i and standard deviation σ_i in the hybrid performance model in simulations
$h_{T,i}$	Input parameter to Algorithm 5.1 of LC i, determines number of sets in partition $\mathcal{Q}_{T,i}$
$H_i(t)$	Auxiliary function for LC i used in regret analysis
$K_i(t)$	Control function, input to Algorithm 5.1 of LC i
l_i	Number of distinct locations visited by the corresponding user from the reduced Gowalla-NY data set in simulations
L	Parameter in Hölder continuity assumption
m_t	Maximum number of workers who should complete task t
$N_{i,q}(t)$	Number of times before task t, in which worker $i \in \mathcal{W}$ was selected to complete a task for exploration purposes when her/his joint context belonged to hypercube $q \in \mathcal{Q}_{T,i}$
$p_i(\mathbf{x}, \mathbf{c})$	Random variable for performance of worker $i \in \mathcal{W}$ with current personal context $\mathbf{x} \in \mathcal{X}_i$ for a task with task context $\mathbf{c} \in \mathcal{C}$
$p_i(\mathbf{x}_{t,i}, \mathbf{c}_t, t)$	Instantaneous performance of worker $i \in \mathcal{W}_t$ with current personal context $\mathbf{x}_{t,i} \in \mathcal{X}_i$ for task t with task context $\mathbf{c}_t \in \mathcal{C}$
$q_i(\mathbf{x}, \mathbf{c})$	Random variable for quality of worker $i \in \mathcal{W}$ with current personal context $\mathbf{x} \in \mathcal{X}_i$ for a task with task context $\mathbf{c} \in \mathcal{C}$
$q_{\max}$	Maximum quality with which a worker can complete a task
$q_{\min}$	Minimum quality with which a worker can complete a task

$q_{t,i}$	Hypercube of partition to which context vector $(\mathbf{x}_{t,i}, \mathbf{c}_t)$ belongs
$\mathcal{Q}_{T,i}$	Partition of the context space of LC i
$R(T)$	Regret of learning with respect to oracle after T rounds
$R_{\text{all}}(T)$	Regret due to select-all-worker phases
$R_{\text{or}}(T)$	Regret due to exploration phases
$R_{\text{oi}}(T)$	Regret due to exploitation phases
$s_{t,j}$	j-th selected worker for task t
$s^*_{t,j}$	j-th best worker for task t
$\mathcal{S}^*$	Collection of optimal subsets of workers for the collection $\{1, ..., T\}$ of tasks
$\mathcal{S}_t$	Set of selected workers for task t
$\mathcal{S}^*_t$	Optimal subset of workers to select for task t
t	Index for task
T	Total number of tasks
$V(t)$	Auxiliary event used in the regret analysis
$w_i\left(x_i^{(2)}\right)$	Location-specific weighting factor in the hybrid performance model in simulations
W	Number of workers
W_t	Number of workers available at the arrival of task t
$W_{\text{ue},t}$	Number of under-explored workers at the arrival of task t
$\mathcal{W}$	Set of workers
$\mathcal{W}_t$	Set of workers available at the arrival of task t
$\mathcal{W}_t^{\text{ue}}$	Set of under-explored workers at the arrival of task t
$x_i^{(1)}$	Worker i's battery state in the hybrid performance model in simulations
$x_i^{(2)}$	Worker i's location in the hybrid performance model in simulations
$\mathbf{x}_{t,i}$	Personal context of worker $i \in \mathcal{W}_t$ at the arrival of task t
$(\mathbf{x}_{t,i}, \mathbf{c}_t)$	Joint (personal and task) context of worker $i \in \mathcal{W}_t$ at the arrival of task t
X_i	Dimension of personal context space of worker $i \in \mathcal{W}$
$\mathcal{X}_i$	Personal context space of worker $i \in \mathcal{W}$
$\mathcal{X}_i \times \mathcal{C}$	Joint (personal and task) context space of worker $i \in \mathcal{W}$
$y_{t,i}$	Binary variable describing if worker $i \in \mathcal{W}_t$ is selected to complete task t
z_i	Parameter for worker i used in regret analysis
α	Parameter in Hölder continuity assumption

γ_i	Parameter for worker i used in regret analysis
$\Gamma_T(A)$	Cumulative worker performance at T achieved by an algorithm A
ϵ	Parameter in ϵ-Greedy algorithm
$\theta_i\left(c, x_i^{(1)}, x_i^{(2)}\right)$	Expected performance of worker i in the hybrid performance model in simulations
$\hat{\theta}_{i,q}(t)$	Estimated performance of worker $i \in \mathcal{W}$ for contexts in hypercube $q \in \mathcal{Q}_{T,i}$ at the arrival of task t
$\theta_i(\mathbf{x}, \mathbf{c})$	Expected performance of worker $i \in \mathcal{W}$ with current personal context $\mathbf{x} \in \mathcal{X}_i$ for a task with task context $\mathbf{c} \in \mathcal{C}$
λ_{AUER}	Input parameter to AUER algorithm in simulations
λ_{HCL}	Factor for control function of Algorithm 5.1 in simulations
λ_{LinUCB}	Input parameter to LinUCB algorithm in simulations
$\nu_i(\mathbf{x}, \mathbf{c})$	Expected quality of worker $i \in \mathcal{W}$ with current personal context $\mathbf{x} \in \mathcal{X}_i$ for a task with task context $\mathbf{c} \in \mathcal{C}$
$\pi_i(\mathbf{x}, \mathbf{c})$	Acceptance rate of worker $i \in \mathcal{W}$ with current personal context $\mathbf{x} \in \mathcal{X}_i$ for a task with task context $\mathbf{c} \in \mathcal{C}$
ρ	Availability probability of each worker in simulations
τ_T	Set of tasks for which $W_t > m_t$, used in regret analysis
τ_T^c	Set of tasks for which HCL enters a select-all-workers phase, used in regret analysis
$\tilde{\tau}_T$	Set of tasks for which HCL enters an exploitation phase, used in regret analysis
$\tilde{\tau}_T^c$	Set of tasks for which HCL enters an exploration phase, used in regret analysis

Bibliography

[ACBF02] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, May 2002.

[AFGM+15] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[AG13] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs," in *Proc. 30th International Conference on International Conference on Machine Learning (ICML) - Vol. 28*, 2013, pp. III–1220–III–1228.

[AKL06] T. Ahmed, K. Kyamakya, and M. Ludwig, "Architecture of a context-aware vertical handover decision model and its performance analysis for GPRS - WiFi handover," in *Proc. IEEE 11th Symposium on Computers and Communications (ISCC)*, 2006, pp. 795–801.

[Alp14] E. Alpaydin, *Introduction to Machine Learning*. The MIT Press, 2014.

[AMS+18] A. Asadi, S. Müller, G. H. Sim, A. Klein, and M. Hollick, "FML: Fast machine learning for 5G mmWave vehicular communications," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2018, pp. 1961–1969.

[ASA+14] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 337–368, 2014.

[ASMK16] H. Al-Shatri, S. Müller, and A. Klein, "Distributed algorithm for energy efficient multi-hop computation offloading," in *Proc. IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[ATvdSB16] S. Amuru, C. Tekin, M. v. der Schaar, and R. M. Buehrer, "Jamming bandits – a novel learning method for optimal jamming," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2792–2808, Apr. 2016.

[AVC11] V. Ambati, S. Vogel, and J. Carbonell, "Towards task recommendation in micro-task markets," in *Proc. 11th AAAI Conference on Human Computation*, 2011, pp. 80–83.

[AvdS16] K. Ahuja and M. van der Schaar, "Dynamic matching and allocation of tasks," *arXiv preprint, arXiv: 1602.02439*, 2016.

[AZDG16] J. G. Andrews, X. Zhang, G. D. Durgin, and A. K. Gupta, "Are we approaching the fundamental limits of wireless network densification?" *IEEE Communications Magazine*, vol. 54, no. 10, pp. 184–190, Oct. 2016.

[BASK18] B. Boiadjieva, H. Al-Shatri, and A. Klein, "Hierarchical beamforming in CRAN using random matrix theory," in *Proc. IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[BBD14a] E. Bastug, M. Bennis, and M. Debbah, "Cache-enabled small cell networks: Modeling and tradeoffs," in *Proc. International Symposium on Wireless Communications Systems (ISWCS)*, 2014, pp. 649–653.

[BBD14b] ——, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[BBZ+15] E. Bastug, M. Bennis, E. Zeydan, M. A. Kader, A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *IEEE Journal of Communications and Networks, Special Issue on Big Data Networking – Challenges and Applications*, vol. 17, no. 6, pp. 549–558, Dec. 2015.

[BC12] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, Dec. 2012.

[BCF+99] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, vol. 1, 1999, pp. 126–134.

[BG14a] P. Blasco and D. Gündüz, "Content-level selective offloading in heterogeneous networks: Multi-armed bandit optimization and regret bounds," *arXiv preprint, arXiv: 1407.6154*, 2014.

[BG14b] ——, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE International Conference on Communications (ICC)*, 2014, pp. 1897–1903.

[BG14c] ——, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2014, pp. 51–55.

[BKK+09] T. Braun, A. Kassler, M. Kihl, V. Rakocevic, V. Siris, and G. Heijenk, "Multihop wireless networks," in *Traffic and QoS Management in Wireless Multimedia Networks: COST 290 Final Report*, V. Siris, T. Braun, F. Barcelo-Arroyo, D. Staehle, G. Giambene, and Y. Koucheryavy, Eds. Boston, MA: Springer US, 2009, pp. 201–265.

[BLS14] A. Badanidiyuru, J. Langford, and A. Slivkins, "Resourceful contextual bandits," in *Proc. 27th Conference on Learning Theory*, 2014, pp. 1109–1134.

[BMZA12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of things," in *Proc. ACM First Edition of the MCC Workshop on Mobile Cloud Computing (MCC)*, 2012, pp. 13–16.

[BSW12] A. Brodersen, S. Scellato, and M. Wattenhofer, "YouTube around the world: Geographic popularity of videos," in *Proc. ACM International Conference on World Wide Web (WWW)*, 2012, pp. 241–250.

[BV04] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[BW16] R. Bhattacharya and E. C. Waymire, "Independence, conditional expectation," in *A Basic Course in Probability Theory*. Cham: Springer International Publishing, 2016, ch. 2, pp. 25–52.

[BWL18] T. E. Bogale, X. Wang, and L. B. Le, "Machine intelligence techniques for next-generation context-aware wireless networks," *ITU Journal: ICT Discoveries, Special Issue 1: The impact of Artificial Intelligence on communication networks and services*, vol. 1, Mar. 2018.

[CFLP16] F. Chiti, R. Fantacci, M. Loreti, and R. Pugliese, "Context-aware wireless mobile autonomic computing and communications: Research trends and emerging applications," *IEEE Wireless Communications*, vol. 23, no. 2, pp. 86–92, Apr. 2016.

[Che15] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[CHH+18] M. Chen, Y. Hao, L. Hu, M. S. Hossain, and A. Ghoneim, "Edge-CoCaCo: Toward joint optimization of computation, caching, and communication on edge cloud," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 21–27, Jun. 2018.

[Chl09] E. Chlebus, "An approximate formula for a partial sum of the divergent p-series," *Applied Mathematics Letters*, vol. 22, no. 5, pp. 732 – 737, May 2009.

[CHMA10] L. B. Chilton, J. J. Horton, R. C. Miller, and S. Azenkot, "Task search in a human computation market," in *Proc. ACM SIGKDD Workshop on Human Computation (HCOMP)*, 2010, pp. 1–9.

[CI97] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," in *Proc. USENIX Symposium on Internet Technologies and Systems*, 1997, pp. 193–206.

[Cis17] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021," Cisco, white paper, Feb. 2017, accessed: 08.11.2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[CLD16] M. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

[CLRS11] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proc. 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 208 – 214.

[CML11] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 1082–1090.

[CNN05] CNN, "Battery life concerns mobile users," Sep. 2005, accessed: 19.11.2018, Report. [Online]. Available: http://edition.cnn.com/2005/TECH/ptech/09/22/phone.study/

[CWY13] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework, results and applications," in *Proc. 30th International Conference on Machine Learning*, ser. Proc. of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. PMLR, 2013, pp. 151–159.

[DA99] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *Proc. 1st International Symposium on Handheld and Ubiquitous Computing (HUC)*. Springer-Verlag, 1999, pp. 304–307.

[DLNW13] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[Dob82] G. Dobson, "Worst-case analysis of greedy heuristics for integer programming with nonnegative data," *Mathematics of Operations Research*, vol. 7, no. 4, pp. 515–531, Nov. 1982.

[DRH11] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the World-Wide Web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, Apr. 2011.

[DS12] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*, 4th ed. Addison-Wesley, 2012.

[EBSLa14] M. ElBamby, M. Bennis, W. Saad, and M. Latva-aho, "Content-aware user clustering and caching in wireless small cell networks," in *Proc. IEEE International Symposium on Wireless Communications Systems (ISWCS)*, 2014, pp. 945–949.

[eMa18] eMarketer, "Mobile time spent 2018," Jun. 2018, accessed: 08.11.2018, Report. [Online]. Available: https://www.emarketer.com/content/mobile-time-spent-2018

[Eva11] D. Evans, "The Internet of things: How the next evolution of the Internet is changing everything," Cisco, San Jose, CA, USA, White paper, 2011, accessed: 05.02.2019. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[FCGS02] M. P. Fromherz, L. S. Crawford, C. Guettier, and Y. Shang, "Distributed adaptive constrained optimization for smart matter systems," in *Proc. AAAI Spring Symposium on Intelligent Embedded and Distributed Systems*, 2002, pp. 34–39.

[FMASK17] M. Fasil, S. Müller, H. Al-Shatri, and A. Klein, "Exploiting caching and cross-layer transitions for content delivery in wireless multihop networks," in *Proc. 2nd Content Caching and Delivery in Wireless Networks Workshop (CCDWN)*, 2017, pp. 1–7.

[FMS+17] A. Frömmgen, S. Müller, S. Sadasivam, A. Klein, A. Buchmann, M. Lehn, and R. Rehner, "Verfahren zur Aufrechterhaltung der Performance einer Multipath-TCP-Verbindung," German Patent DE102 015 114 164 (A1), 2017.

[Fou10] J.-C. Fournier, *Graph Theory and Applications*. ISTE, 2010.

[Fré04] A. Fréville, "The multidimensional 0–1 knapsack problem: An overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1 – 21, May 2004.

[FSK+18] P. Felka, A. Sterz, K. Keller, B. Freisleben, and O. Hinz, "The context matters: Predicting the number of in-game actions using traces of mobile augmented reality games," in *Proc. ACM 17th International Conference on Mobile and Ubiquitous Multimedia (MUM)*, 2018, pp. 25–35.

[FSM+15] A. Frömmgen, S. Sadasivam, S. Müller, A. Klein, and A. Buchmann, "Poster: Use your senses: A smooth multipath TCP WiFi/mobile handover," in *Proc. 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2015, pp. 248–250.

[FTH16] C. Funai, C. Tapparello, and W. Heinzelman, "Mobile to mobile computational offloading in multi-hop cooperative networks," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–7.

[GALM07] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube traffic characterization: A view from the edge," in *Proc. ACM Conference on Internet Measurement (IMC)*, 2007, pp. 15–28.

[GBMY97] D. Goodman, J. Borras, N. B. Mandayam, and R. Yates, "Infostations: A new system model for data and messaging services," in *Proc. IEEE Vehicular Technology Conference (VTC)*, vol. 2, 1997, pp. 969–973.

[GEE+16] 5G-PPP, ERTICO, EFFRA, EUTC, NEM, CONTINUA, and Networld2020 ETP, "5G empowering vertical industries," Feb. 2016, accessed: 27.01.2019, Brochure. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2016/02/BROCHURE_5PPP_BAT2_PL.pdf

[GLZ14] C. Gentile, S. Li, and G. Zappella, "Online clustering of bandits," in *Proc. 31st International Conference on Machine Learning (ICML)*, 2014, pp. II–757–II–765.

[GMDC13] N. Golrezaei, A. Molisch, A. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, Apr. 2013.

[Gol05] A. Goldsmith, *Wireless Communications*. New York, NY, USA: Cambridge University Press, 2005.

[GP85] B. Gavish and H. Pirkul, "Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality," *Mathematical Programming*, vol. 31, no. 1, pp. 78–105, Jan. 1985.

[GR14] I. Gradshteyn and I. Ryzhik, "0 - introduction," in *Table of Integrals, Series, and Products*, 8th ed., D. Zwillinger and V. Moll, Eds. Boston: Academic Press, 2014, pp. 1 – 24.

[Gro03] GroupLens, "MovieLens 1M Dataset," Feb. 2003, accessed: 06.08.2016. [Online]. Available: http://grouplens.org/datasets/movielens/1m/

[GS14] D. Geiger and M. Schader, "Personalized task recommendation in crowdsourcing information systems – Current state of the art," *Decision Support Systems*, vol. 65, pp. 3 – 16, Sep. 2014.

[GSM18] GSMA Intelligence, "The mobile economy 2018," Feb. 2018, accessed: 08.11.2018, Report. [Online]. Available: https://www.gsmaintelligence.com/research/2018/02/the-mobile-economy-2018/660/

[Gur15] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

[GWG+16] Y. Gong, L. Wei, Y. Guo, C. Zhang, and Y. Fang, "Optimal task recommendation for mobile crowdsourcing with privacy control," *IEEE Internet Things Journal*, vol. 3, no. 5, pp. 745–756, Oct. 2016.

[GZQL12] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design*, 2012, pp. 279–284.

[Hen03] K. Henricksen, "A framework for context-aware pervasive computing applications," Ph.D. dissertation, Computer Science, School of Information Technology and Electrical Engineering, The University of Queensland, 2003.

[HGI18] K. Heßler, T. Gschwind, and S. Irnich, "Stabilized branch-and-price algorithms for vector packing problems," *European Journal of Operational Research*, vol. 271, no. 2, pp. 401 – 419, Dec. 2018.

[HK15] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, Dec. 2015.

[HL05] C. A. Hoffner and K. J. Levine, "Enjoyment of mediated fright and violence: A meta-analysis," *Media Psychology*, vol. 7, no. 2, pp. 207–237, Nov. 2005.

[Hoe63] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, Mar. 1963.

[HPS+15] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing – a key technology towards 5G," European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, white paper, Sep. 2015, accessed: 26.01.2019. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[HRR+18] R. Hans, B. Richerzhagen, A. Rizk, U. Lampe, R. Steinmetz, S. Klos née Müller, and A. Klein, "Little boxes: A dynamic optimization approach for enhanced cloud infrastructures," in *Proc. European Conference on Service-Oriented and Cloud Computing (ESOCC)*, 2018, pp. 199–206.

[HV12] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *Proc. 26th AAAI Conference on Artificial Intelligence*, 2012, pp. 45–51.

[HWN12] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[HYH+16] R. Huo, F. R. Yu, T. Huang, R. Xie, J. Liu, V. C. M. Leung, and Y. Liu, "Software defined networking, caching, and computing for green wireless networks," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 185–193, Nov. 2016.

[HZL16] K. Han, C. Zhang, and J. Luo, "Taming the uncertainty: Budget limited robust crowdsensing through online learning," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1462–1475, Jun. 2016.

[IR02] A. L. Iacono and C. Rose, "Infostations: New perspectives on wireless data networks," in *Next Generation Wireless Networks*, S. Tekinay, Ed. Boston, MA: Springer US, 2002, ch. 1, pp. 3–63.

[JHF03] R. Jansen, S. Hanemann, and B. Freisleben, "Proactive distance-vector multipath routing for wireless ad hoc networks," in *Proc. Communication Systems and Networks (CSN)*, 2003.

[JZR+17] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[KAC+15] M. J. Kochenderfer, C. Amato, G. Chowdhary, J. P. How, H. J. D. Reynolds, J. R. Thornton, P. A. Torres-Carrasquillo, N. K. Üre, and J. Vian, *Decision Making Under Uncertainty: Theory and Application*, 1st ed. The MIT Press, 2015.

[Kar84] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. Sixteenth Annual ACM Symposium on Theory of Computing (STOC)*, 1984, pp. 302–311.

[KASK19] K. Kiekenap, H. Al-Shatri, and A. Klein, "Trade-off between measurement accuracy and quantization precision for minimum Bayes risk in wireless networked control systems," in *Proc. 12th International ITG Conference on Systems, Communications and Coding (SCC)*, 2019, pp. 1–6.

[KB97] M. Kahani and H. W. P. Beadle, "Decentralised approaches for network management," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 36–47, Jul. 1997.

[KL10] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[KLAC03] B.-J. Ko, K.-W. Lee, K. Amiri, and S. Calo, "Scalable service differentiation in a shared storage cache," in *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2003, pp. 184–193.

[KLJ+10] S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, and J. Song, "A scalable and energy-efficient context monitoring framework for mobile personal sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 686–702, May 2010.

[KLLB13] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[KNMS10] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma, "Regret bounds for sleeping experts and bandits," *Machine Learning*, vol. 80, no. 2-3, pp. 245–272, Sep. 2010.

[KNY09] M. Katoozian, K. Navaie, and H. Yanikomeroglu, "Utility-based adaptive radio resource allocation in OFDM wireless networks with traffic prioritization," *IEEE Transactions on Wireless Communications*, vol. 8, no. 1, pp. 66–71, Jan. 2009.

[KP00] V. Kuleshov and D. Precup, "Algorithms for the multi-armed bandit problem," *Journal of Machine Learning Research*, vol. 1, pp. 1–48, Oct. 2000.

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[KS12] L. Kazemi and C. Shahabi, "Geocrowd: Enabling query answering with spatial crowdsourcing," in *Proc. 20th ACM Int. Conf. on Advances in Geographic Information Systems (SIGSPATIAL)*, 2012, pp. 189–198.

[KTvK18] S. Klos née Müller, C. Tekin, M. van der Schaar, and A. Klein, "Context-aware hierarchical online learning for performance maximization in mobile crowdsourcing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1334–1347, Jun. 2018.

[LAK17] H. Q. Le, H. Al-Shatri, and A. Klein, "Optimal joint power allocation and task splitting in wireless distributed computing," in *Proc. 11th International ITG Conference on Systems, Communications and Coding (SCC)*, 2017, pp. 1–6.

[LAS04] Y. Lu, T. F. Abdelzaher, and A. Saxena, "Design, implementation, and evaluation of differentiated caching services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, pp. 440–452, May 2004.

[LCLS10] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th ACM International Conference on World Wide Web (WWW)*, 2010, pp. 661–670.

[LCQ16] H. Liu, Z. Chen, and L. Qian, "The three primary colors of mobile systems," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 15–21, Sep. 2016.

[LH00] Y.-D. Lin and Y.-C. Hsu, "Multihop cellular: A new architecture for wireless communications," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, vol. 3, 2000, pp. 1273–1282.

[LK03] Y. Liu and E. Knightly, "Opportunistic fair scheduling over multiple wireless channels," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, vol. 2, 2003, pp. 1106–1115.

[LL17] Y. Liu and M. Liu, "An online learning approach to improving the quality of crowd-sourcing," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2166–2179, Aug. 2017.

[LMZL16] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 1451–1455.

[LPP10] T. Lu, D. Pal, and M. Pal, "Contextual multi-armed bandits," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 485–492.

[LR85] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4–22, Mar. 1985.

[LTZ+18] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-driven mobile edge caching placement for adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965–984, Apr. 2018.

[Lun92] J. Lunze, *Feedback Control of Large Scale Systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1992.

[LXvdSL16a] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.

[LXvdSL16b] ——, "Popularity-driven content caching," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2016, pp. 1–9.

[LZ07] J. Langford and T. Zhang, "The epoch-greedy algorithm for contextual multi-armed bandits," in *Proc. 20th International Conference on Neural Information Processing Systems*, 2007, pp. 817–824.

[MAN14] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[MASW+15] S. Müller, H. Al-Shatri, M. Wichtlhuber, D. Hausheer, and A. Klein, "Computation offloading in wireless multi-hop networks: Energy minimization via multi-dimensional knapsack problem," in *Proc. IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2015, pp. 1717–1722.

[MAvK16] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *Proc. IEEE International Conference on Communications (ICC)*, 2016, pp. 1–7.

[MAvK17] ——, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

[MBASK18] T. Mahn, D. Becker, H. Al-Shatri, and A. Klein, "A distributed algorithm for multi-stage computation offloading," in *Proc. IEEE 7th International Conference on Cloud Networking (CloudNet)*, 2018, pp. 1–6.

[MF11] D. Minarolli and B. Freisleben, "Utility-based resource allocation for virtual machines in cloud computing," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, 2011, pp. 410–417.

[MF14] ——, "Distributed resource allocation to virtual machines via artificial neural networks," in *Proc. 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2014, pp. 490–499.

[MH16] S. Maghsudi and E. Hossain, "Multi-armed bandits with application to 5G small cells," *IEEE Wireless Communications*, vol. 23, no. 3, pp. 64–73, Jun. 2016.

[MMAS+16] M. Mousavi, S. Müller, H. Al-Shatri, B. Freisleben, and A. Klein, "Multi-hop data dissemination with selfish nodes: Optimal decision and fair cost allocation based on the Shapley value," in *Proc. IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[MMF17] A. Mazrekaj, D. Minarolli, and B. Freisleben, "Distributed resource allocation in cloud computing using multi-agent systems," *Telfor Journal*, vol. 9, no. 2, pp. 110–115, 2017.

[MN10] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conference on Hot Topics in Cloud Computing*, 2010, pp. 4–4.

[MS10] M.-L. Mares and Y. Sun, "The multiple meanings of age for television content preferences," *Human Communication Research*, vol. 36, no. 3, pp. 372–396, Jul. 2010.

[MSS13] P. Makris, D. N. Skoutas, and C. Skianis, "A survey on context-aware mobile and wireless networking: On networking and computing environments' integration," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 362–386, First Quarter 2013.

[MYM02] D. Maniezzo, K. Yao, and G. Mazzini, "Energetic trade-off between computing and communication resource in multimedia surveillance sensor network," in *Proc. 4th International Workshop on Mobile and Wireless Communications Network*, 2002, pp. 373–376.

[MYZ+17] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[NMAS+18] D. Nowak, T. Mahn, H. Al-Shatri, A. Schwartz, and A. Klein, "A generalized Nash game for mobile edge computation offloading," in *Proc. 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2018, pp. 95–102.

[NN94] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Methods in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[PAM10] E. Patouni, N. Alonistioti, and L. Merakos, "Modeling and performance evaluation of reconfiguration decision making in heterogeneous radio network environments," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 4, pp. 1887–1900, May 2010.

[PCA07] S. Pandey, D. Chakrabarti, and D. Agarwal, "Multi-armed bandit problems with dependent arms," in *Proc. 24th ACM International Conference on Machine Learning (ICML)*, 2007, pp. 721–728.

[PDG+16] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5G era: Enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, Mar. 2016.

[PIAT14] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2014, pp. 1078–1086.

[PIST16] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2995–3007, Apr. 2016.

[PT13] K. Poularakis and L. Tassiulas, "Exploiting user mobility for wireless content delivery," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2013, pp. 1017–1021.

[PWLP15] M. Peng, C. Wang, V. Lau, and H. V. Poor, "Fronthaul-constrained cloud radio access networks: insights and challenges," *IEEE Wireless Communications*, vol. 22, no. 2, pp. 152–160, Apr. 2015.

[PZCG14] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, First Quarter 2014.

[QCZ14] L. Qin, S. Chen, and X. Zhu, "Contextual combinatorial bandit and its application on diversified online recommendation," in *Proc. SIAM International Conference on Data Mining*, 2014, pp. 461–469.

[Qua18] Qualcomm Technologies, Inc., "VR and AR pushing connectivity limits," Oct. 2018, accessed: 13.06.2019, Report. [Online]. Available: https://www.qualcomm.com/media/documents/files/vr-and-ar-pushing-connectivity-limits.pdf

[RGZ11] P. J. Rentfrow, L. R. Goldberg, and R. Zilca, "Listening, watching, and reading: The structure and correlates of entertainment preferences," *Journal of Personality*, vol. 79, no. 2, pp. 223–258, Apr. 2011.

[Rob52] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, Sep. 1952.

[RP03] P. Rong and M. Pedram, "Extending the lifetime of a network of battery-powered mobile devices by remote processing: A Markovian decision-based approach," in *Proc. 40th ACM Annual Design Automation Conference*, 2003, pp. 906–911.

[RT99] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, Apr. 1999.

[RZZS15] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "Exploiting mobile crowdsourcing for pervasive cloud services: Challenges and solutions," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 98–105, Mar. 2015.

[SAT+14] A. Sengupta, S. Amuru, R. Tandon, R. Buehrer, and T. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. IEEE International Symposium on Wireless Communications Systems (ISWCS)*, 2014, pp. 917–921.

[SB98] R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[SC17] M. Safran and D. Che, "Real-time recommendation algorithms for crowdsourcing systems," *Applied Computing and Informatics*, vol. 13, no. 1, pp. 47 – 56, Jan. 2017.

[Sch86] A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1986.

[Sch95] W. N. Schilit, "A system architecture for context-aware mobile computing," Ph.D. dissertation, Columbia University, New York, NY, USA, 1995.

[SGD+13] K. Shanmugam, N. Golrezaei, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[SHP+14] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proc. 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2014, pp. 287–296.

[SHT02] B. N. Schilit, D. M. Hilbert, and J. Trevor, "Context-aware communication," *IEEE Wireless Communications*, vol. 9, no. 5, pp. 46–54, Oct. 2002.

[SKA+18] G. Sim, S. Klos, A. Asadi, A. Klein, and M. Hollick, "An online context-aware machine learning algorithm for 5G mmWave vehicular communications," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2487–2500, Dec. 2018.

[Sli14] A. Slivkins, "Contextual bandits with similarity information," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2533–2568, Jan. 2014.

[SMF+12] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, 2012, pp. 59–66.

[SNHH15] L. Song, D. Niyato, Z. Han, and E. Hossain, *Wireless Device-to-Device Communications and Networks*. New York, NY, USA: Cambridge University Press, 2015.

[SRI+15] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jäntti, and A. Ghosh, "Machine-type communications: Current status and future perspectives toward 5G systems," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 10–17, Sep. 2015.

[STv16] C. Shen, C. Tekin, and M. van der Schaar, "A non-stochastic learning approach to energy efficient mobility management," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3854–3868, Dec. 2016.

[SV13] A. Slivkins and J. W. Vaughan, "Online decision making in crowdsourcing markets: Theoretical challenges," *ACM SIGecom Exchanges*, vol. 12, no. 2, pp. 4–23, Dec. 2013.

[SWKC12] H.-Y. Shi, W.-L. Wang, N.-M. Kwok, and S.-Y. Chen, "Game theory for wireless sensor networks: A survey," *Sensors*, vol. 12, no. 7, pp. 9055–9097, 2012.

[TF09] M. Tahir and R. Farrell, "Optimal communication-computation tradeoff for wireless multimedia sensor network lifetime maximization," in *Proc. IEEE Wireless Communications and Networking Conference*, 2009, pp. 1–6.

[TGFS17] H. To, G. Ghinita, L. Fan, and C. Shahabi, "Differentially private location protection for worker datasets in spatial crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 934–949, Apr. 2017.

[TSK15] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Transactions on Spatial Algorithms and Systems*, vol. 1, no. 1, pp. 2:1–2:28, Jul. 2015.

[TTSRJ14] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings, "Efficient crowdsourcing of unknown experts using bounded multi-armed bandits," *Artificial Intelligence*, vol. 214, pp. 89 – 111, Sep. 2014.

[TV05] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. New York, NY, USA: Cambridge University Press, 2005.

[TvdS15a] C. Tekin and M. van der Schaar, "Distributed online learning via cooperative contextual bandits," *IEEE Transactions on Signal Processing*, vol. 63, no. 14, pp. 3700–3714, Mar. 2015.

[TvdS15b] ——, "RELEAF: An algorithm for learning and exploiting relevance," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 716–727, Jun. 2015.

[TZvdS14] C. Tekin, S. Zhang, and M. van der Schaar, "Distributed online learning in social recommender systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 638–652, Aug. 2014.

[uHC14] U. ul Hassan and E. Curry, "A multi-armed bandit approach to online spatial task assignment," in *Proc. 11th IEEE International Conference on Ubiquitous Intelligence and Computing (UTC)*, 2014, pp. 212–219.

[VM05] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Proc. European Conference on Machine Learning (ECML)*. Springer-Verlag, 2005, pp. 437–448.

[WCT+14] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[WDM+15] M. Wichtlhuber, S. Dargutev, S. Müller, A. Klein, and D. Hausheer, "QTrade: A quality of experience based peercasting trading scheme," in *Proc. IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2015, pp. 1–10.

[WHY+18] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 7–38, Firstquarter 2018.

[WZL12] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2012, pp. 2716–2720.

[WZZ+17] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[XLL07] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *Proc. 13th International Conference on Parallel and Distributed Systems*, 2007, pp. 1–8.

[You16] YouGov, "Few smartphone users interested in phones without headphone jacks," Sep. 2016, accessed: 19.11.2018, Report. [Online]. Available: https://today.yougov.com/topics/consumer/articles-reports/2016/09/22/smartphone-longer-battery-life-headphone-jack

[You18] ——, "Smartphone users still want long-lasting batteries more than shatterproof screens," Feb. 2018,

accessed: 19.11.2018, Report. [Online]. Available: https://today.yougov.com/topics/technology/articles-reports/2018/02/20/smartphone-users-still-want-longer-battery-life

[YZvdS18] J. Yoon, W. R. Zame, and M. van der Schaar, "Deep sensing: Active sensing using multi-directional recurrent neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2018, pp. 1–19.

[ZC17] L. Zheng and L. Chen, "Maximizing acceptance in rejection-aware spatial crowdsourcing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1943–1956, Sep. 2017.

[ZGC+14] C. Zhou, Y. Guo, Y. Chen, X. Nie, and W. Zhu, "Characterizing user watching behavior and video quality in mobile devices," in *Proc. IEEE International Conference on Computer Communication and Networks (ICCCN)*, 2014, pp. 1–6.

[ZH16] Y. Zhao and Q. Han, "Spatial crowdsourcing: Current state and future directions," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 102–107, Jul. 2016.

[Zil88] D. Zillmann, "Mood management through communication choices," *American Behavioral Scientist*, vol. 31, no. 3, pp. 327–340, Jan. 1988.

[ZJZ10] M. Zekri, B. Jouaber, and D. Zeghlache, "Context aware vertical handover decision making in heterogeneous wireless networks," in *Proc. IEEE Local Computer Network Conference*, 2010, pp. 764–768.

[ZSGK09] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network – measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501–514, Mar. 2009.

Author's Publications and Patents

[AMS+18] A. Asadi, S. Müller, G. H. Sim, A. Klein, and M. Hollick, "FML: Fast machine learning for 5G mmWave vehicular communications," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2018, pp. 1961–1969.

[ASMK16] H. Al-Shatri, S. Müller, and A. Klein, "Distributed algorithm for energy efficient multi-hop computation offloading," in *Proc. IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[FHK+16] A. Frömmgen, M. Hassan, R. Kluge, M. Mousavi, M. Mühlhäuser, S. Müller, M. Schnee, M. Stein, and M. Weckesser, "Mechanism transitions: A new paradigm for a highly adaptive internet," Technical Report, Darmstadt, Mar. 2016. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/5370/

[FMASK17] M. Fasil, S. Müller, H. Al-Shatri, and A. Klein, "Exploiting caching and cross-layer transitions for content delivery in wireless multihop networks," in *Proc. 2nd Content Caching and Delivery in Wireless Networks Workshop (CCDWN)*, 2017, pp. 1–7.

[FMS+17] A. Frömmgen, S. Müller, S. Sadasivam, A. Klein, A. Buchmann, M. Lehn, and R. Rehner, "Verfahren zur Aufrechterhaltung der Performance einer Multipath-TCP-Verbindung," German Patent DE102 015 114 164 (A1), 2017.

[FSM+15] A. Frömmgen, S. Sadasivam, S. Müller, A. Klein, and A. Buchmann, "Poster: Use your senses: A smooth multipath TCP WiFi/mobile handover," in *Proc. 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2015, pp. 248–250.

[HRR+18] R. Hans, B. Richerzhagen, A. Rizk, U. Lampe, R. Steinmetz, S. Klos née Müller, and A. Klein, "Little boxes: A dynamic optimization approach for enhanced cloud infrastructures," in *Proc. European Conference on Service-Oriented and Cloud Computing (ESOCC)*, 2018, pp. 199–206.

[KTvK18] S. Klos née Müller, C. Tekin, M. van der Schaar, and A. Klein, "Context-aware hierarchical online learning for performance maximization in mobile crowdsourcing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1334–1347, Jun. 2018.

[MASW+15] S. Müller, H. Al-Shatri, M. Wichtlhuber, D. Hausheer, and A. Klein, "Computation offloading in wireless multi-hop networks: Energy minimization via multi-dimensional knapsack problem," in *Proc. IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2015, pp. 1717–1722.

[MAvK16] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in

Proc. IEEE International Conference on Communications (ICC), 2016, pp. 1–7.

[MAvK17] ——, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

[MMAS+16] M. Mousavi, S. Müller, H. Al-Shatri, B. Freisleben, and A. Klein, "Multihop data dissemination with selfish nodes: Optimal decision and fair cost allocation based on the Shapley value," in *Proc. IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[SKA+18] G. Sim, S. Klos, A. Asadi, A. Klein, and M. Hollick, "An online context-aware machine learning algorithm for 5G mmWave vehicular communications," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2487–2500, Dec. 2018.

[WDM+15] M. Wichtlhuber, S. Dargutev, S. Müller, A. Klein, and D. Hausheer, "QTrade: A quality of experience based peercasting trading scheme," in *Proc. IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2015, pp. 1–10.

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO
Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO
Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde.

§9 Abs. 1 PromO
Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO
Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Datum und Unterschrift